2

ERIM 208600-1-T
SHD-88-190

Technical Report

REMOTE MINEFIELD DETECTION SYSTEM (REMIDS) REAL-TIME PROCESSING ARCHITECTURE STUDY

J.N. MUCZYNSKI
R.J. HORNER
P.L. MOHAN
J.A. SALINGER

NOVEMBER 1988

Report for Period July 1988—November 1988

Performed under Contract N00014-84-C-0493 funded by Office of Naval Research, Arlington, Virginia

Submitted to:

Environmental Lab
Waterways Experiment Station
U.S. Army Corps of Engineers
P.O. Box 631
Vicksburg, Mississippi 39180-0631

Attn: Mr. R. A. Goodson

ERIM P.O. Box 8618
Ann Arbor, MI 48107-8618

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Unlimited Distribution
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBERS(S) 208600-1-T	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION ERIM — 6b. OFFICE SYMBOL (if applicable)	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) P.O. Box 8618 Ann Arbor, MI 48107	7b. ADDRESS (City, State, and ZIP Code) 800 N. Quincy St. Arlington, VA 22217
8a. NAME OF FUNDING /SPONSORING ORGANIZATION WES-Corps of Engineers — 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-84-C-0443
8c. ADDRESS (City, State, and ZIP Code) P.O. Box 631 Vicksburg, MS 39180-0631	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)
REmote MInefield Detection Systems Real-Time Processing Architecture Study (U)

12. PERSONAL AUTHOR(S)
J. N. Muczynski, R.J. Horner, P. L. Mohan, J. A. Salinger

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM 7/88 TO 9/88	1988 October 06	68

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

A heterogeneous multi-processor system is to be integrated for real time implementation of a multi-modal image processing algorithm for mine field detection. Available off-the-shelf multi-processor systems, including the AISI 5000, Datacube Max-Video Boards, Transputers, Hypercubes, ITI Series 150, AMT DAP510 and multiple 68020 CPUs, were considered for implementation of various parts of the algorithms. The alternatives were evaluated using performance, cost, and support criteria. A system combining Datacube Max-Video Boards and Transputers is recommended. Keywords: Infrared detectors; REMIDS (Remote Minefield Detection System). (EG)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) — 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

Table of Contents

Table of Contents . . . iii

List of Figures . . . v

List of Tables . . . vii

List of Acronyms . . . ix

1.0 INTRODUCTION . . . 1

1.1 Background . . . 2

1.2 Evaluation Methodology . . . 4

1.3 Summary of Results . . . 5

2.0 DATA COMPRESSION PROCESSOR . . . 7

2.1 Data Compression Algorithm Analysis . . . 7

2.2 Data Compression Hardware Performance Analysis . . . 9

2.2.1 Massively Parallel, Mesh Connected Architectures . . . 9

2.2.2 Pipelined, Recirculating Architectures . . . 13

2.3 Data Compression Processor Acceptance Criteria . . . 13

2.4 Data Compression Processor Analysis Results . . . 16

2.4.1 Datacube MaxVideo Boards . . . 18

2.4.2 AISI AISI5000 . . . 22

2.4.3 ITI Board Set . . . 23

3.0 CLASSIFICATION PROCESSOR . . . 25

3.1 Classification Algorithm Analysis . . . 25

3.1.1 Clustering Algorithm Analysis . . . 27

3.1.2 Linearity and Density Screening (LDS) Algorithm Analysis . . . 31

3.1.3 Classification Processing Load . . . 34

3.1.4 Fixed-Point Precision Requirements . . . 36

3.2 Classification Hardware Performance Analysis . . . 38

3.2.1 AMT DAP500 . . . 39

3.2.1.1 Fixed-Point Performance Analysis . . . 39

3.2.1.2 Floating-Point Performance Analysis . . . 44

3.2.2 Parsytec T800 Transputer . . . 46

Table of Contents

3.2.2.1 T800 Processor Capabilities 48

3.2.2.2 T800 Classification Performance 50

3.2.3 N-Cube N-Cube7 51

3.2.4 Force 68020 CPU Board Set 54

3.2.4.1 Force Board Capabilities 54

3.2.4.2 Force Board Classification Performance 56

3.2.5 AISI AISI5000 57

3.3 Classification Processor Acceptance Criteria 58

3.4 Classification Processor Analysis Results 60

3.4.1 AMT DAP500 61

3.4.2 Parsytec T800 Transputer 66

3.4.3 N-Cube N-Cube7 66

3.4.4 Force 68020 Board Set 67

3.4.5 AISI AISI5000 67

4.0 PROPOSED HARDWARE CONFIGURATION 69

4.1 Operational Sequencing 69

4.2 Open Issues 72

List of Figures

Figure 1	Minefield Detection Algorithm	3
Figure 2	Generic Data Compression Processor	8
Figure 3	AISI5000 System	11
Figure 4	Organization of the AISI5000 Processor Elements	12
Figure 5	Typical Programmable Pipeline Module Image Processor Configuration	14
Figure 6	Two-Stage Pipeline-based Data Compression Processor	21
Figure 7	Core Clustering Algorithm	28
Figure 8	Linearity and Density Screening Algorithm	32
Figure 9	DAP500 System	40
Figure 10	DAP500 Processor Element Organization	41
Figure 11	Transputer-based Classification System	47
Figure 12	T800 Transputer Chip Organization	49
Figure 13	Complete N-Cube10 System	52
Figure 14	N-Cube Processor Chip Organization	53
Figure 15	68020-based Classification System	55
Figure 16	REMIDS II Airborne Processor	70
Figure 17	Execution Sequence Through the Data Compression Processor	71

List of Tables

Table 1 Data Compression Hardware Alternative Assessment 17

Table 2 Data Compression Throughput Analysis (DataCube Boardset) 19

Table 3 Data Compression Throughput Analysis (AISI5000) 20

Table 4 Decision Analysis Worksheet (DAP500/N-Cube) 62

Table 5 Decision Analysis Worksheet (Parsytec/Force) 63

Table 6 Decision Analysis Worksheet (AISI5000) 64

List of Acronyms

AISI	Applied Intelligent Systems Inc.
AMT	Active Memory Technology
CISC	Complex Instruction Set Computer A processing unit whose instructions take more than one clock cycle of time for execution (see RISC).
CPU	Central Processing Unit
DAP	Distributed Array Processor A series of AMT SIMD processors.
DSP	Digital Signal Processing
ERIM	Environmental Research Institute of Michigan
FIFO	First-In First-Out Memory device in which the first data placed into the input side of the device will be the first data out of the output side of the device.
FIR	Finite Impulse Response A type of digital filter.
Hz	Hertz Cycles per cycle.
IEEE	Institute of Electical and Electronics Engineers
I/O	Input/Output (for example, I/O device)
IR	InfraRed Refers to spectral wavelengths outside the visible spectrum and longer than visible red.
ITI	Imaging Technology Inc.
kB	kiloBytes 2^{10} Bytes
LDS	Linearity and Density Screening
MIMD	Multiple Instruction Multiple Data A digital hardware architecture classification.

List of Acronyms

MIPS	Million Instructions Per Second
MFLOPS	Million Floating-Point Operations Per Second
NIR	Near InfraRed Refers to spectral wavelengths just outside the visible spectrum and longer than visible red.
PE	Processing Element
RAM	Random Access Memory
REMIDS	REmote MInefield Detection System
RGB	Red, Green, Blue Refers to a video transmission format in which the red, green and blue signals of a color picture are transmitted separately (i.e., on separate cables). Also refers to various equipment that supports this transmission scheme.
RISC	Reduced Instruction Set Computer A processing unit capable of executing one instruction per clock cycle.
ROM	Read Only Memory
SIMD	Single Instruction Multiple Data A digital hardware architecture classification.
VCR	Video Cassette Recorder
VLSI	Very Large Scale Integration
VME	Versabus Module Europe A European-accepted Versabus heritage module.
WES	Waterways Experimental Station

1.0 INTRODUCTION

This document reports the architecture study results for the real-time implementation of remote minefield detection system (REMIDS) algorithms using images from the REMIDS II sensor. The real-time REMIDS system consists of two intercommunicating processors. The processes that they perform are referred to as data compression and classification.

Sections 2.0 and 3.0 and their related subsections discuss the data compression and classification algorithms of the REMIDS II System.

Sections 2.1 and 3.1 discuss the data compression and classification algorithms. Sections 2.2 and 3.2 discuss the performance of the processors that were considered for performing the data compression and classification algorithms.

Sections 2.3, 2.4, 3.3 and 3.4 determine a quantitative score for each of the data compression and classification processors. These scores are determined through a two-part procedure (explained in Section 1.2) and are used to evaluate the processors executing the data compression and classification algorithms.

Section 4.0 discusses the proposed hardware configuration for the REMIDS II system and lists the unresolved issues.

Throughout this report the following terms are used. *Architecture* refers to the organizational structure of a processor. *Processor* refers to an implementation of an architecture. More than one implementation of an architecture may exist. *System* refers to an assembly of *processors* that form a whole—in this case, the REMIDS system. *Operation* refers to the actions performed when an instruction is executed. Examples are the addition and multiplication operations of scalar arithmetic; the convolution, thresholding and feature extraction operations of image analysis; and the dilation and erosion operations of morphological image processing. *Algorithm* refers to a set of operations and rules for getting a specific output from a specific input. *Process* refers to an algorithm that changes the general form of data in a definable way. Consequently, data compression, clustering, and linearity and density screening are processes and algorithms since they have image, vector, scalar, and morphological data types. *Architecture overhead* refers to the processing power required for operations that do not contribute to the processing of algorithm data, such as I/O management, processor synchronization, etc. Architecture overhead also includes processing power that is wasted when part of the hardware is unused because the hardware is not perfectly fitted to the algorithm. For instance, during synchronization of multiple parallel processors, some processors will idle—contributing nothing to performance—while waiting for other processors to complete their operations. *Architecture head room* refers to the available safety margin of processing power. Although some processing power may be unused, it is not included as part of the architecture head room if it is not able to be used (for example, idle time during processor synchronization).

The following sections provide background material on the sensor and the algorithms for mine detection, describe the methodology used to evaluate the data compression and classification processors, and summarize the results of the study.

1.1 Background

The remote minefield detection system (REMIDS) developed by the Waterways Experimentation Station (WES) uses a sensor, which includes active and passive modalities, to form images for remote sensing of anti-personnel and anti-armor mines. The sensor includes a thermal infrared (IR) detector and two (parallel and cross) polarization sensitive near infrared (NIR) detectors with an active polarized IR laser illuminator. The sensor is intended to be mounted in an autonomous or remotely piloted air vehicle that will traverse navigable land areas to locate and identify the boundaries of minefields.

The algorithms use an adaptive multi-sensor fusion approach to finding the minefields (see Figure 1, Minefield Detection Algorithm). The polarization information is formed into two images: the difference between the parallel and cross polarization returns, and the total of the these two returns. These, plus the thermal image, make up the three image inputs to the minefield detection algorithm.

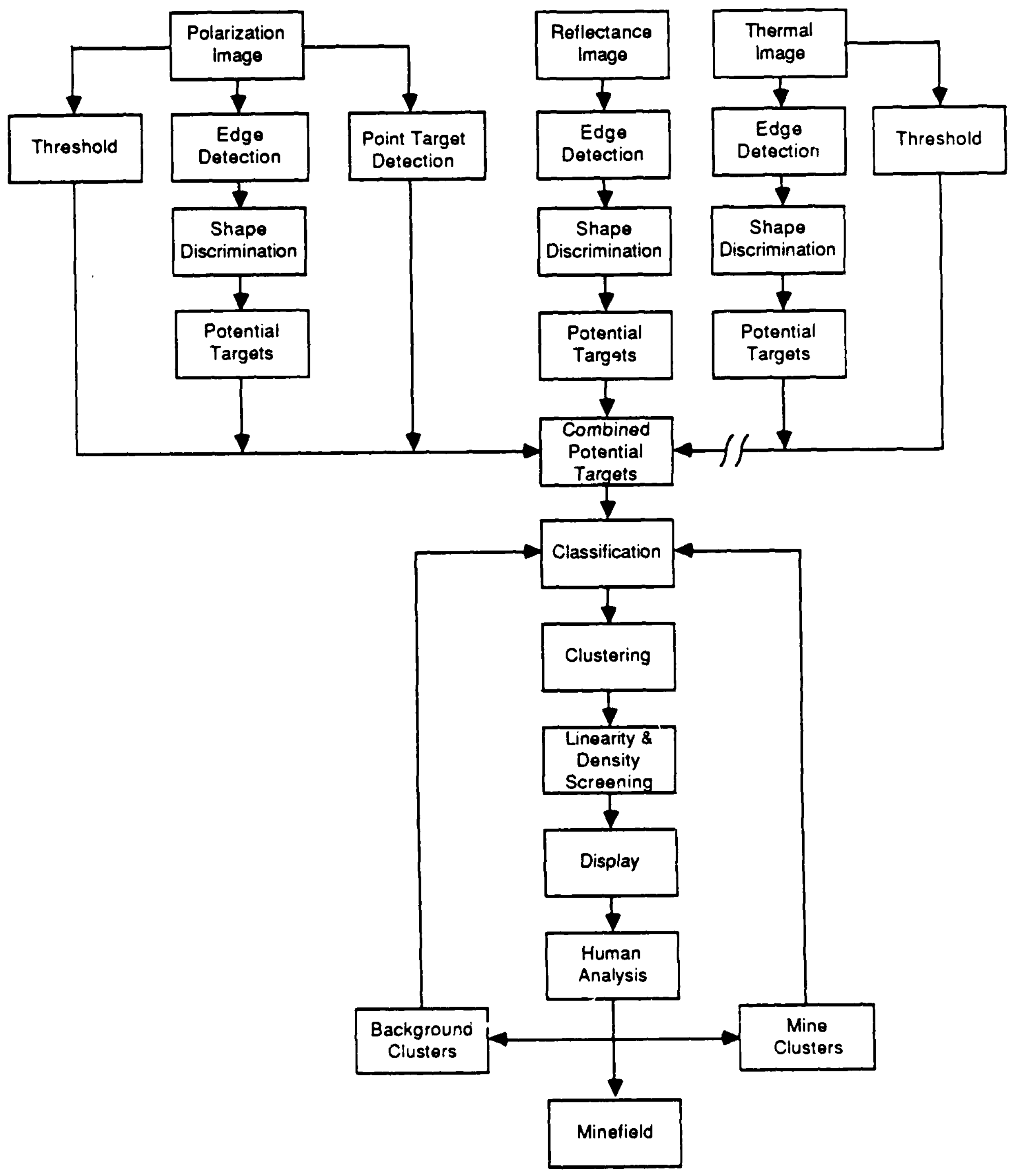

Figure 1. Minefield Detection Algorithm

The algorithms can be divided into two major sections. The top half of Figure 1 (the part above the classification block) shows an adaptive data compression algorithm that, with slight variations, is executed on each of the three image types: polarized NIR detectors' difference, polarized NIR detectors' total, and IR thermal detector. This algorithm determines those pixels that are likely to be part of a mine. Pixels are chosen based on an analysis of edges in the region of the pixel. The lower half of Figure 1 depicts an adaptive classification algorithm that categorizes the pixels screened in the data compression algorithm into individual clusters and then regroups them into objects. These objects can then be analyzed for shape, size, and spatial distributions to determine if they are part of a minefield. The classification algorithm is explained in detail in Section 3.1.

1.2 Evaluation Methodology

Alternative synthesis and evaluation has been constrained, primarily, by time and funding. In order to provide the most thorough analysis practical in the time available, several individuals were assigned to evaluate different architectures for the data compression and classification processors. To provide consistent evaluation of the architectures and to limit the impact of biases, a consistent procedure of evaluation is used. The procedure of choosing processors has two major steps. These steps are applied to evaluate the data compression processors in Sections 2.3 and 2.4, and the classification processors in Sections 3.3 and 3.4.

In the first step, a set of decision criteria and priorities are established. The decision criteria are divided into a mandatory and a desirable set. The mandatory criteria are essential for success of the real-time project. The desirable criteria are assigned a weight, on a scale of one to ten, that reflects their importance for successful completion of the program. The most important desirable criteria receive a weight of ten. Other desirable criteria are given weights in proportion to the most important desirable criteria.

The second step is to assign a score to each processor for each desirable criteria, on a scale of one to ten, that reflects how well each processor satisfies each of the desirable criteria; each processor is also studied to determine if each mandatory criteria has been satisfied. If a particular architecture does not satisfy a mandatory criteria, then it is eliminated from further consideration. The processor that best satisfies the desirable criteria receives a score of 10. Other processors are given scores proportional to a score of 10. Whenever possible, the scores are based on quantitative information. Each processor then receives a composite score—computed by summing the products of the criteria weights and the processor scores for each criteria—that is used to justify the choice of processor for the REMIDS system.

1.3 Summary of Results

Three alternatives were evaluated for real-time execution of the Data Compression algorithms, the AISI AISI5000, the ITI Series 150 board set, and the Datacube Max-Video board set. It was concluded that the Datacube board set provides the best balance of cost, performance, and support for this algorithm. The alternatives either had significant risk of not being able to execute the algorithms in real time or required development of special interfaces to handle the line lengths from the REMIDS II sensor and the required high resolution display. The Datacube board set provides the required interfaces, modules to execute each of the sections of the algorithm in real time, and reasonable software and field support.

Five alternatives were considered for real-time execution of the Classification Processor algorithms. These were the AMT DAP500, the N-Cube N-Cube7, the Parsytec Transputer boards, the AISI AISI5000, and multiple 68020 boards from Force. It was concluded that the Parsytec Transputer boards provide the best balance of cost, performance, and support for these algorithms. It is anticipated that 13 Transputers will provide the required processing power. The system is expandable with boards that contain two transputers each. The support software is well established and the cost is well below any of the alternatives.

Section 2.0 explains in greater detail why the Datacube board set was chosen for integrating and executing the Data Compression algorithm. Section 3.0 explains why the Parsytec Transputer board set was chosen for integrating and executing the Classification algorithm.

2.0 DATA COMPRESSION PROCESSOR

The data compression algorithms require extensive processing for each pixel acquired by the sensor. Two approaches exist for implementing the algorithms in real time. Since the algorithms were initially implemented for non-real-time execution, we assumed a frame-by-frame approach. An alternative, which must be considered, is to implement the algorithms in a line-by-line approach. The principal difference between these approaches is in the adaptive thresholding aspects of the algorithm. The line-by-line approach provides a smoother transition when the characteristics of the data change, but it will also require more computation. For this evaluation, it is assumed that the existing frame-by-frame approach will be followed. However, one of the desirable decision criteria is the availability of enough flexibility and computational power to implement a line-by-line approach.

An overview of the data compression algorithm and hardware performance analyses can be found in Sections 2.1 and 2.2. The acceptance criteria and evaluation results corresponding to the two steps of the evaluation process, explained in Section 1.2, are contained in Sections 2.3 and 2.4.

2.1 Data Compression Algorithm Analysis

Execution of the data compression algorithms can be performed by the generic processor depicted in Figure 2, Generic Data Compression Processor. Three image channels—sum, difference, and thermal—are shown passing through the processor in parallel. The hardware resources may be shared across multiple channels and operate upon each channel sequentially. The classes of operations required by the data compression algorithms can be categorized as follows:

- Input Data Formatting
- Image Buffering
- Histogramming and Image Statistic Computation
- Convolution
- Thresholding
- Feature Extraction
- Image Display

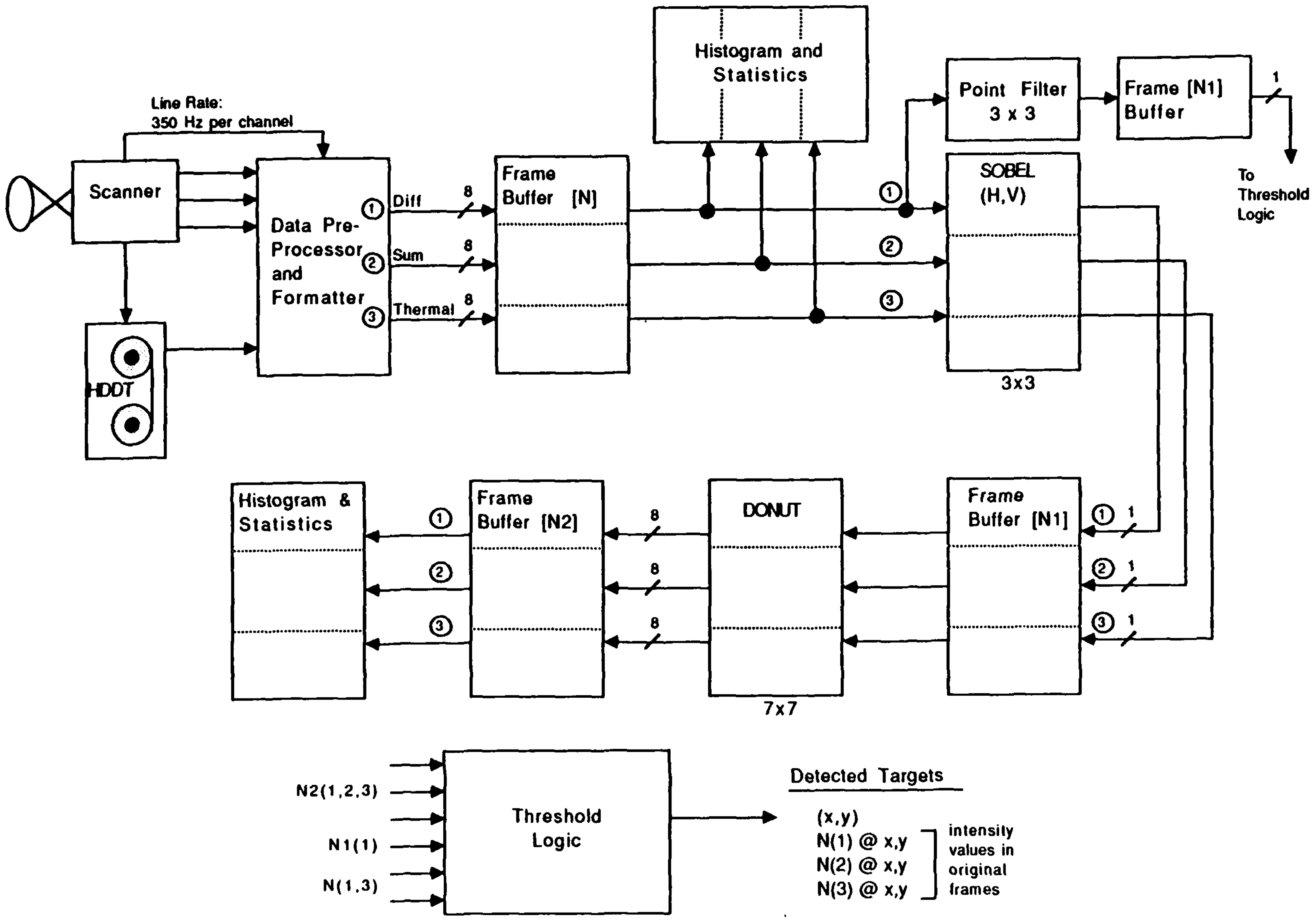

Figure 2. Generic Data Compression Processor

These operations form a required minimum set of functions that any potential data compression processor must support.

2.2 Data Compression Hardware Performance Analysis

There are two general classes of processing architectures that can provide enough computational power to execute the data compression algorithms in real time: massively parallel and pipelined. These two parallel processing architectures were considered in developing an implementation of the data compression algorithms.

In the fine-grained, massively parallel architecture, many simple processing elements (PEs) execute a common program in unison. Each PE has local storage, as well as access to the storage of PEs in a local neighborhood. Each PE executes the algorithm on a pixel or group of pixels. This architecture is discussed in Section 2.2.1.

The PE of the pipelined architecture transfers data in a raster-scan format through a series of processing stages. Each stage executes one or several instructions from the algorithm on the data that it receives. The current stage then sends the results to the next stage for execution of the next set of instructions while it processes the next set of input pixels. The pipeline of stages acts as a production line. Each stage executes a portion of the algorithm and passes the results to the next PE. This architecture is discussed in Section 2.2.2.

2.2.1 Massively Parallel, Mesh Connected Architectures

The first architecture that was considered is the massively parallel architecture. The Applied Intelligent Systems AISI5000 parallel processor is the implementation studied. It is a single instruction multiple data (SIMD) architecture consisting of an $N \times 1$ dimensional array of computational elements; the array size can range from 128×1 to 1024×1. The AISI5000 array is as wide as the image because there is one PE allocated for each column in the image. Enough memory is coupled directly to each PE to hold the entire column of the image data, as well as to provide a large amount of storage space for intermediate results. A single image operation will load instructions into the PEs, read rows of data from the associated memory to the PEs, perform the transformation, and then write the resulting transformed rows back into memory. Rows are read and written sequentially until the entire image is processed.

The AISI5000 system consists of three major components:

- A general purpose 68000 host processor and peripherals,
- Special purpose high speed input/output hardware for the parallel processor, and
- An SIMD array parallel processor and controller.

Figure 3, AISI5000 System, shows the relationship between the host, input/output (I/O), and parallel processor components. The 68000 host includes memory and peripherals; it has a conventional design and programming methodology. The I/O sections transfer data in and out of the memory associated with the PEs and contain corner-turning logic. The corner-turning logic serially buffers the data of a complete image line, then passes the entire row into the processing array memories in parallel. The data destined for the parallel memory within the PEs can come from a number of sources, including cameras and standard 68000 addressable random access memory (RAM). The data coming from parallel memory can go to a number of destinations, including video monitors and the 68000 RAM. The parallel processing portion of the AISI5000 consists of the PEs, their associated memory, and the controller that directs their function.

The linear array can contain up to 1024 programmable, bit-serial PEs; each is tightly coupled to its own local single-bit-wide memory and has connections to two neighboring PEs as shown in Figure 4, Organization of the AISI5000 Processor Elements. Each PE has direct access to 32kbits of RAM within its own column. The data in all adjacent memory columns can be accessed by a PE through connections to its two neighboring PEs. The PEs can perform three distinct types of functions: Boolean, neighborhood, and arithmetic. These functions are performed on single bits read from the memory; the results of the functions are written back to memory. A three-channel I/O system facilitates movement of data in and out of the parallel memory. The I/O operates independently of, and asynchronously with, the PEs.

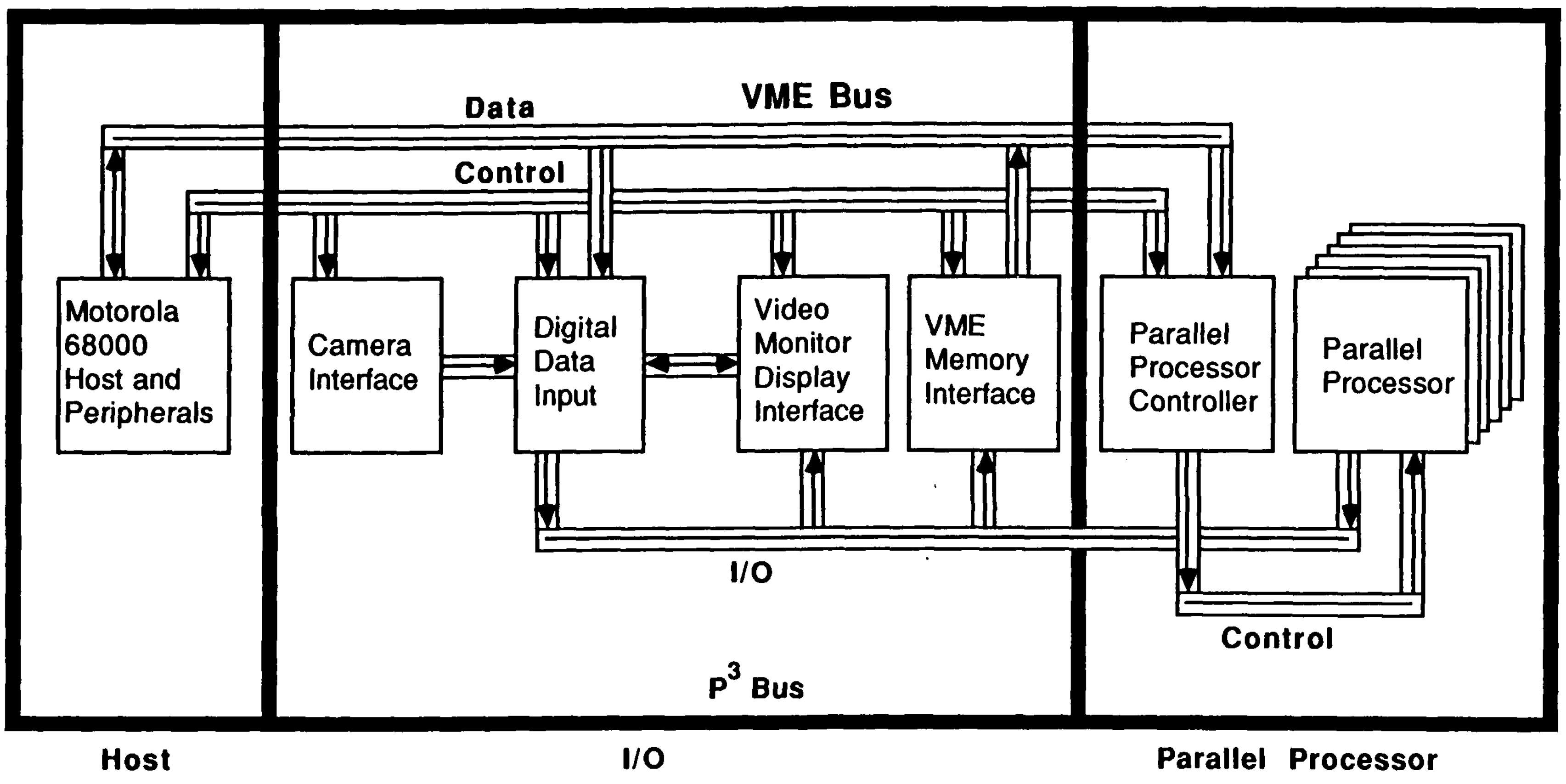

Figure 3. AISI 5000 System

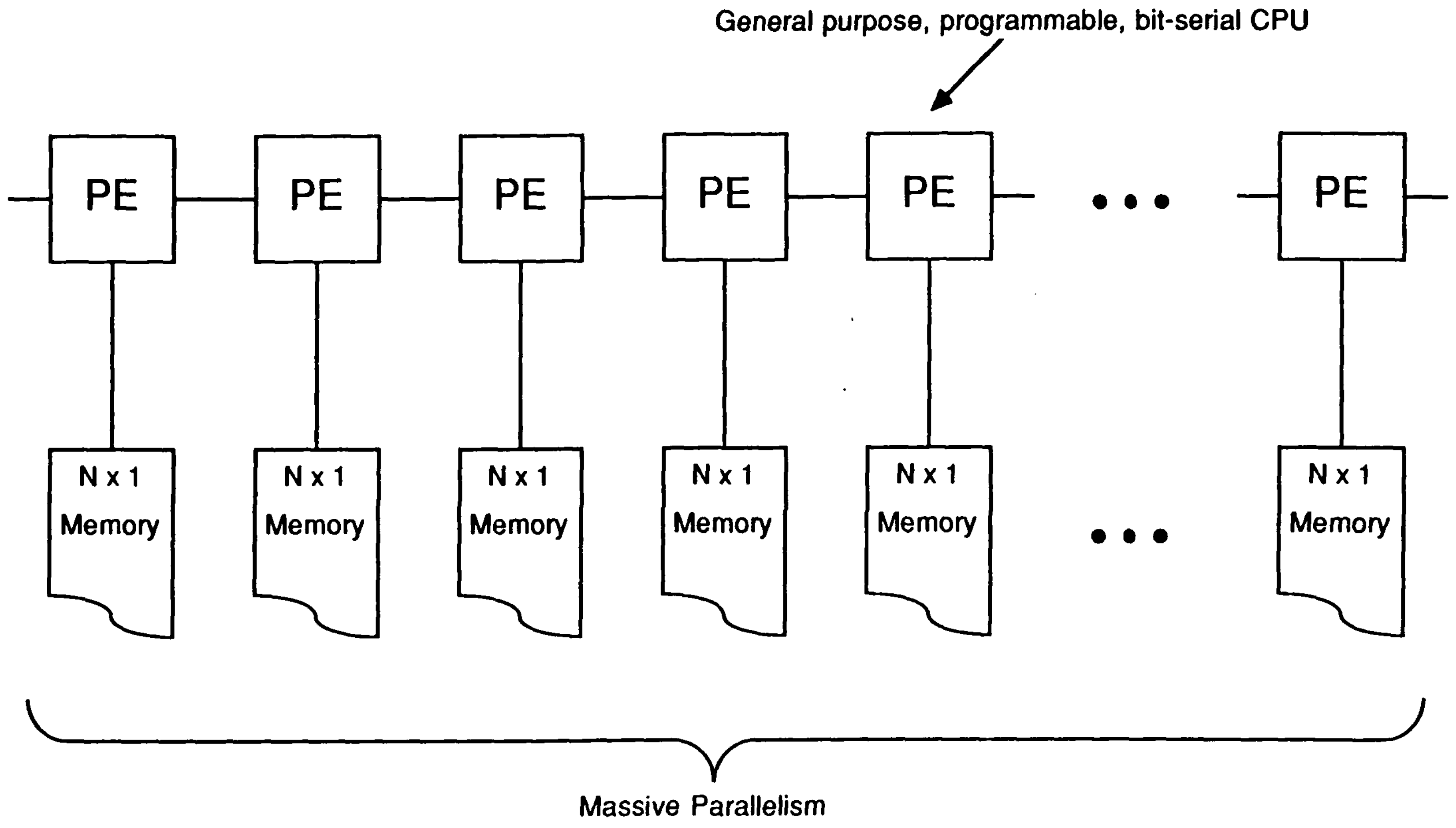

Figure 4. Organization of the AISI 5000 Processor Elements

2.2.2 Pipelined, Recirculating Architectures

The second architecture that was analyzed is the pipelined architecture. With a modular family of programmable, pipelined processing units, a real-time image processor can be customized for a specific algorithm. For the pipelined architecture, the Datacube and ITI processors were considered. They can be customized to a set of algorithms by distributing the various operations required for image processing and analysis to separate Versabus Module Europe (VME) boards that have been designed to perform the operations of image algorithms. Each board supports a powerful, flexible set of operations of a particular class. Partitioning of operations to distinct hardware boards typically occurs along the following boundaries:

- Image Acquisition/Display
- Image Memory
- Convolution/Finite Impulse Response (FIR) Filtering
- Histogram/Feature Extraction
- General Digital Signal Processing (DSP)
- Neighborhood Processing

These modules may be arranged to support a number of parallel architectures, from multi-pass recirculating architectures to fully pipelined architectures. The processing units are designed to allow pixel data to flow synchronously through high-speed pipelines at video rates—typically 10MHz (10 Million Hertz or 10 Million cycles per second). Multiple dedicated image data buses are supported throughout a multi-board configuration. For recirculating implementations, the routing of these buses and the interconnection of various processing modules can be modified on-the-fly through cross-point switching units. A separate 68000-based host processor provides overall control and updates control/configuration registers on the image processing modules. A pipelined configuration is shown in Figure 5, Typical Programmable Pipeline Module Image Processor Configuration.

2.3 Data Compression Processor Acceptance Criteria

The first step of the evaluation process, explained in Section 1.2, is applied here for the data compression hardware. Based on the data compression algorithms currently implemented

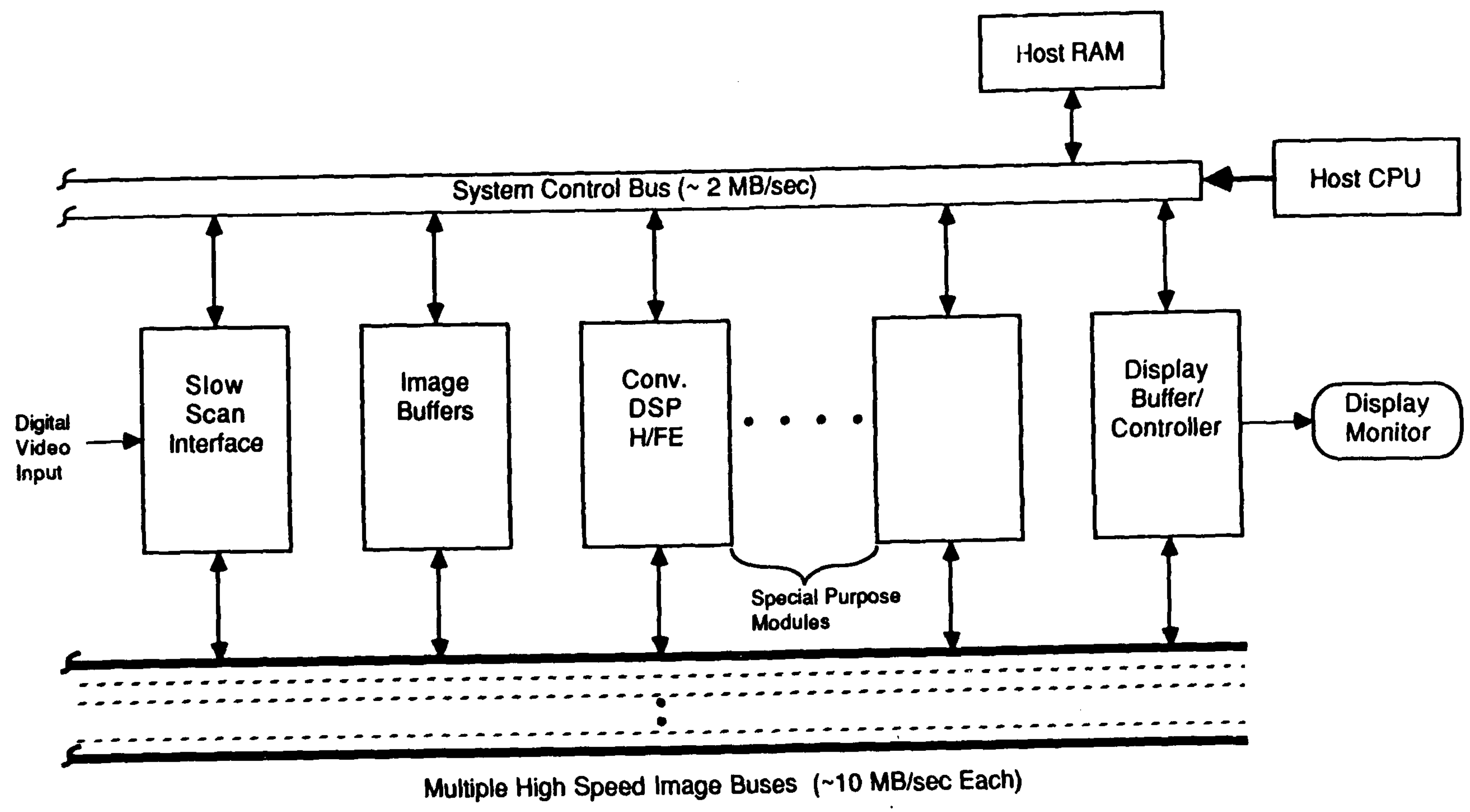

Figure 5. Typical Programmable Pipeline Module Image Processor Configuration

and considering real-time execution constraints, the following mandatory and desirable hardware selection criteria were established:

MANDATORY

- 700 pixels per scan line
- High-resolution (700+ element) display support
- Algorithm functionality
- Established product
- Processing power above established minimum
- Size, weight, and power below established maximums

DESIRED

- Algorithm overhead (9)
- Expandability/Configurability (7)
- Architecture overhead (7)
- Significant number of units in field (5)
- Quality of field support (4)
- Slow scan digital input support (3)
- Display function support (3)
- Minimum cost (2)
- Minimal size, weight, and power (2)

The mandatory items are essential to the completion of the real-time REMIDS project. The desirable items are given weights from one to ten—shown here in parentheses—that reflect the ease or difficulty of integrating the individual hardware and algorithms into a real-time system within the current development constraints.

The available algorithm overhead is the most important of the desirable criteria. This criteria refers to the ability of the family of processor modules to accommodate the various algorithm operations called for in the data compression algorithms. The sponsor prefers these operations to be implemented as directly as possible with a minimum of software development—reconstruction of operations through sets of primitive commands—and with as few recirculations of the images as possible.

Expandability and architecture overhead are also desirable criteria. Easy expandability may be critical if the data compression algorithms require more processing power than we have estimated. Small increments are more desirable for economic reasons, but reducing the additional programming effort—required when expanding the processor size— should have a higher priority. Architecture overhead refers to the processing power required for operations that do not contribute to the processing of algorithm data, such as I/O management, processor synchronization, etc. Architecture overhead also includes processing power that is wasted when part of the hardware is unused because the hardware is not perfectly fitted to the algorithm. For instance, during synchronization of multiple parallel processors, some processors will idle—contributing nothing to performance—while waiting for other processors to complete their operations. The programmer may also have difficulty utilizing the hardware to full capacity if the development system is inadequate.

The sponsor prefers a slow-scan digital interface. The non-standard scanner video data rate requires that pixels be clocked into the processor at sensor-driven rates. A hardware module that will accommodate these requirements will eliminate the need for designing and building custom interface hardware and software.

Image display functions are a key component of the data compression processor. For instance, overlay features are also highly desirable and will simplify the task of superimposing flagged pixels (mines) upon the original image. Thus, we prefer the board set to include digital image display hardware and software support features. In addition, the system requires support for high-resolution displays because the image line lengths exceed standard video formats.

The following section applies these criteria to the individual architectures considered.

2.4 Data Compression Processor Analysis Results

Table 1, Data Compression Processor Analysis, summarizes the AISI, Datacube, and ITI processor qualities based on the criteria described in Section 2.3. Determining whether a processor possesses adequate processing power is a difficult criteria to evaluate. Besides raw peak processing rate (for example, the number of 3×3 convolutions per second), overhead such as host I/O, module reprogramming, bus contentions, etc., must also be considered in

Table 1. Data Compression Hardware Alternative Assessment

CRITERIA		AISI-5000		Datacube		ITI Series 150	
REQUIRED							
> 512 Line Lengths		YES		Some Functions		Must Mosaic	
Hi-Res Display Support		YES		YES		Future (TBD)	
Functionality		With Software Developement		YES		YES	
Compatible Bus		YES (VME)		YES (VME)		YES (VME)	
Established Product		YES		YES		YES	
Processing Power		Multiple Units Required *		YES		YES	
Size, Weight, Power		4.5 cubic feet 530 watts		2 cubic feet 20 Slots		2 cubic feet 20 Slots	
DESIRED	(Weight)						
Slow Scan Input Capability	[3]	Custom Design Required	3	DC - 10 MHz 1 - 4096 Pixs.	10	Beta Testing	5
Display Support	[3]	1K x 1K	10	2K x 1K	10	Future Product	2
Field Support	[4]		7	Overnight	10	Overnight	10
Units in Field	[5]	40 Units	8		10		10
Cost	[2]	$70,000 - $80,000	6	$50,000	10		8
Size, weight, Power	[2]		8		9		9
Expandability	[7]	1024 PE's Maximum	8	Open Ended Architecture	9		7
Software Devel. Support	[10]	C library, Unix	8	C library, Sun Unix	8	Itex 151, Sun Unix	8
Architecture Overhead	[7]		6		8		8
Algorithm Overhead	[9]		10		7		6
SCORE			403		450		384

* Given 'literal' translation of existing algorithms

establishing accurate estimates of net throughput. Optimization techniques that could be applied in both hardware and software can provide significant performance gains, but they are difficult to predict.

A throughput analysis for the AISI and the Datacube systems is summarized in Table 2, Data Compression Throughput Analysis (Datacube Board Set), and Table 3, Data Compression Throughput Analysis (AISI5000) .

We divided the existing data compression algorithms into a set of generic operations (histogram, convolution, threshold, etc.) that could be mapped to the hardware architectures being studied. The processing time for each operation was then estimated based on manufacturer's benchmarks for identical or related operations. A preliminary estimate of processor performance is possible when given the required input frame rate, knowing the sequence of generic operations that must be performed per frame, and knowing their respective execution times on a given set of hardware. Adequate time margins must be available to accommodate the overhead not considered in this simplified analysis.

The following sections discuss the evaluation of the Datacube, AISI5000, and ITI hardware for their ability to perform the data compression algorithms at real-time rates.

2.4.1 Datacube MaxVideo Boards

The second step of the evaluation process, explained in Section 1.2, is applied here to the Datacube MaxVideo boards. The Datacube Maxvideo board set is particularly well matched to the operations specified in the current data compression algorithms. Its two-stage pipelined processor—each stage consisting of multiple processing boards—should provide more than adequate timing margins for processing and overhead operations. Figure 6, Two-Stage Pipeline-based Data Compression Processor, depicts a two-stage pipeline processor for implementing the data compression algorithms. For those module functions that do not support processing of variable regions of interest or of line lengths greater than 512 pixels (for example, histogram/feature extraction), two passes over the image frame will be

Table 2. Data Compression Throughput Analysis (Datacube Board Set)

CONDITIONS :

Single Processor Shared Over Three Channels
700 Pixel by 176 Lines Per Frame
Three Frames Every .5 Second (6 Frames Per Second)
10MHz Pixel Processing Rate
Two-stage Pipeline
One Second Processing Latency
Algorithms Executed "Literally"

Algorithm Operation	Passes Through Frames	Processing Time (mSec)
STAGE I		
Frame Copy	3	37
Histogram/Statistics	3	99
Point Filter/Threshold	2	24.6
(3x3 Convolution)		
(Threshold)		
Sobel Magnitude	6	74
(3x3 Convolution)		
(Mag. Detect)		
(Threshold)		
		234.6
STAGE II		
Donut Filter	3	37
(7x7 Convolution)		
Histogram/Statistics	3	78
Threshold	11	136
(6 Compares)		
(5 Boolean)		
Feature Extract	4	50
		301

Note: Processing overhead times must be factored in and will decrease resulting timing margins.

TIMING MARGINS :

Stage I	500 - 235 = 265 mSec (53%)
Stage II	500 - 301 = 199 mSec (40%)

Table 3. Data Compression Throughput Analysis (AISI 5000)

<u>CONDITIONS</u>:

Single Processor Shared Over Three Channels
700 Pixel by 176 Lines Per Frame
Three Frames Every .5 Second (6 Frames Per Second)
Algorithms Executed "Literally"

Algorithm Operation	Processing Time (mSec)
Load Frames	18
Histogram/Statistics	2400
Point Filter/Threshold (3x3 Convolution) (Threshold)	11
Sobel Magnitude (3x3 Convolution) (Mag.Detect) (Threshold)	33.6
Donut Filter (7x7 Convolution)	90
Threshold (6 Compares) (5 Boolean)	8
Feature Extraction	61
	2621.6

Note: Processing overhead times must be factored in and will decrease resulting timing margins.

<u>TIMING MARGIN</u>:
No Margin

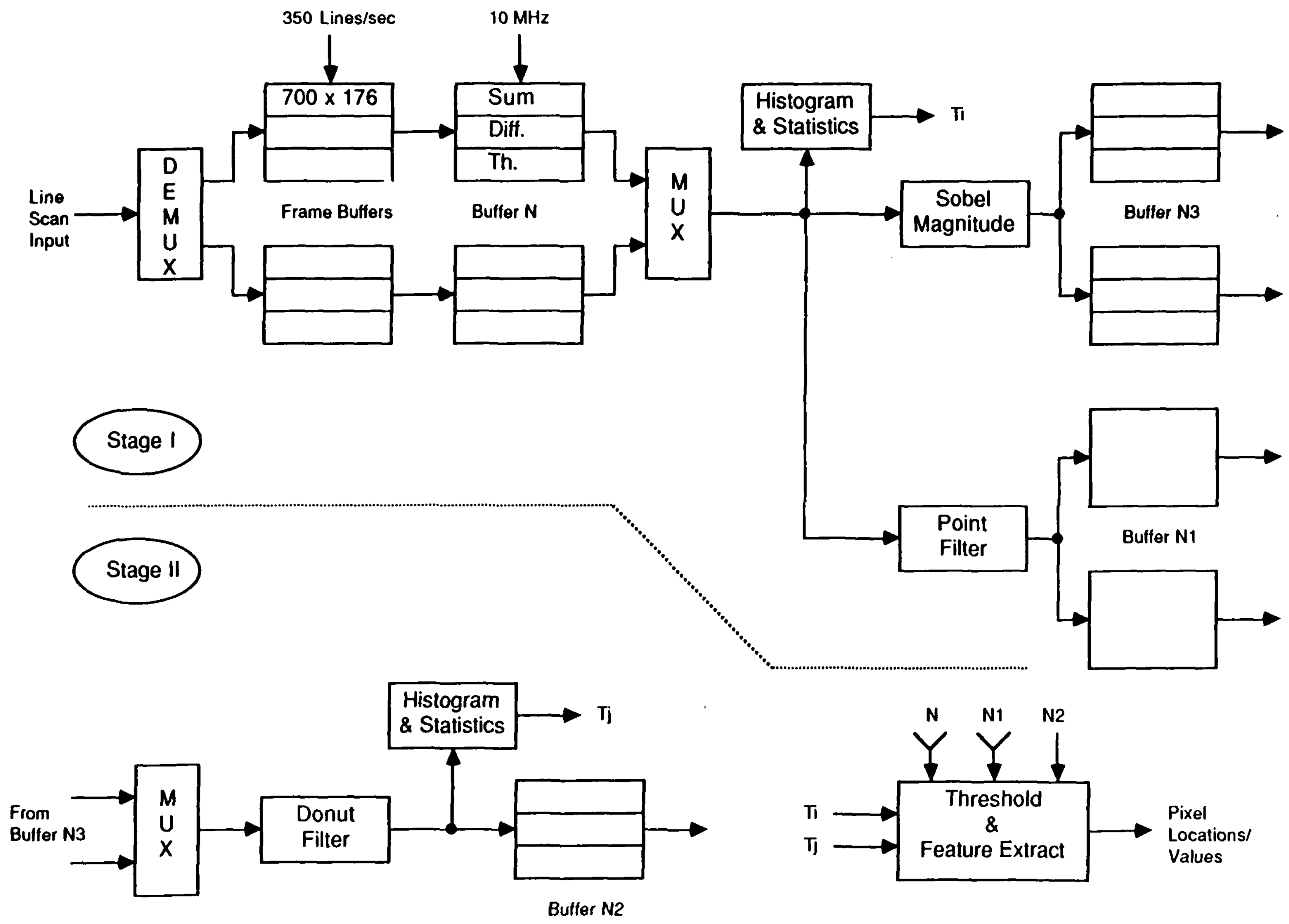

Figure 6. Two-Stage Pipeline-based Data Compression Processor

required. The results are then combined. However, it may prove more efficient to perform the histogram function using Datacube's Euclid DSP module, rather than wait for the Feature Max module to cycle through a full 512×512 frame. Because this board requires a 512×512 image and the REMIDS image is one-third this size, 336 lines of the image will not contain data. The Feature Max board must process 512 lines per pass; this is a restriction not well suited to REMIDS images (176 lines per frame).

Considering that the data compression algorithms are essentially fully developed and will require only minor parameter adjustments in the future, the Datacube implementation will require the least amount of effort in implementing the algorithms in hardware. Thus, the Datacube board set provides a very attractive performance/cost quotient. Some degree of flexibility is available in the sequencing of operations and the interconnecting of processing elements, particularly if we use the cross-point switch module. However, overhead times in the algorithm operations and PE synchronization can be expected to rapidly increase as conditional branching increases.

2.4.2 AISI AISI5000

The second step of the analysis process, explained in Section 1.2, is applied here to the AISI5000. A single 700 PE AISI processor—one PE per image column—is not suited to direct implementation of the WES data compression algorithms as currently specified. This incompatablity results primarily from the very slow rate at which multiple image histograms can be computed (0.4 seconds per 700×176 frame). The AISI performance on the other required algorithm operations is more than adequate. The AISI processor works well with algorithm implementations that employ image morphology rather than histogramming to perform adaptive background normalization. The data compression algorithm could be reworked to take advantage of the operations suited to the AISI5000 architecture and remain functionally equivalent to the existing algorithm. If the data compression algorithms were reworked, the AISI5000 could meet the performance requirements for a real-time data compression architecture.

If evaluating different data compression algorithms—optimized for extended capability, robustness, etc.—was desirable, the relatively constrained architecture using dedicated processing boards becomes much less desirable and the flexibility of the AISI linear array architecture would be more attractive. Given the resources to remap existing algorithms to the AISI system (employing some fundamentally different thinking but achieving the same results), one can achieve a hardware architecture with much more flexibility in terms of pixel precision, conditional execution, and frame sizes. This flexibility is particularly evident with programmable algorithm operators, such as nonlinear filtering, shape analysis, etc. However, the data compression algorithm analysis and selection procedure has already been completed. The advantages of flexibility are outweighed by the ability of the Datacube board set to map almost directly to the selected algorithms.

2.4.3 ITI Board Set

Because the ITI board set does not support a high-resolution display format, it has been eliminated from further consideration.

3.0 CLASSIFICATION PROCESSOR

An evaluation of the classification processors, considered for implementing the algorithms shown in the lower half of Figure 1, Minefield Detection System, is discussed here. The classification processor accepts pixels from the data compression processor and selects those pixels belonging to minefields. The classification processor performs high-level computations that are qualitatively different from the image computations performed in the data compression algorithm shown in the upper half of Figure 1, above the classification block. Consequently, a more flexible, general purpose architecture is required. The following sections describe the classification algorithm and processing load, the candidate classification architectures and processors, and the processor evaluation results.

3.1 Classification Algorithm Analysis

This section describes the Environmental Research Institute of Michigan's (ERIM) understanding of the classification algorithm and determines the processing power required for its real-time execution. The analysis is based on an examination of the source code supplied to ERIM by WES.

The classification algorithm consists of a clustering algorithm followed by a linearity and density screening (LDS) algorithm. Pixels emerging from these algorithms are tagged with a value corresponding to the likelihood that they belong to a mine in a minefield. The (x, y) position of the pixel and its tag are used to highlight the potential minefields in one of the original infrared (IR) images of the terrain. Operator feedback on the accuracy of the highlighted objects can then be used to refine algorithm parameters during a survey flight.

The first of the two classification algorithms, the clustering algorithm, receives pixels selected by the data compression processor that are likely to belong to mines. The pixels are three radiance values associated with the same (x, y) position; these pixels are the thermal IR, and the sum and difference of the two polarized NIR images. The clustering algorithm is based on the principal that the pixels of mines will produce similar trios of radiance values and that clustering of values with similar radiance trios will segregate mine pixels from background pixels.

To identify clusters, the three radiance values are treated as orthogonal vectors in a three-dimensional space and thus specify a single point in space (denoted $\mathbb{R}^3$). Clusters are simply groups of all pixel location trios within a specified distance of one another as determined by a scaled distance formula. The scaled distance—a variation of the conventional square-root sum-of-squares distance—is defined in Section 3.1.1. Clusters are represented as the mean location of all the points in the cluster. The clustering algorithm analysis is described in detail in Section 3.1.1.

A new input point is compared to all other clusters by calculating the point's scaled distance to all existing clusters and sorting these distances to find the closest cluster. If the scaled distance to the closest cluster is within defined limits, the algorithm merges the point into the cluster. This causes the cluster position, which is the average of all the points contained in the cluster, to shift. Therefore, it must be repeatedly compared to all other clusters and merged as long as the new position is less than a threshold distance from another cluster. The merge occurs once for 99% of all clusters containing a single point. If the single point cluster is not close enough to another cluster, then it remains a cluster with one point until another cluster is close enough to be merged with it. When the algorithm merges a pixel with a cluster, or merges two clusters, the new mean center is calculated according to Equation (2) in Section 3.1.1. In either case, the point's cluster is passed to the LDS algorithm.

The second of the two classification algorithms, the LDS algorithm, examines all the points from each cluster contained in a single frame of imagery to determine the likelihood that a cluster corresponds to a minefield. The (x, y) coordinates in the original image of each of the points contained in a cluster are examined and the points are grouped into individual objects. Objects are groups of pixels that are screened and received from the data compression algorithm, are parts of an identical cluster, and are sufficiently close together in the terrain image to be considered the same terrain object. A linear regression, applied to the center of mass of these objects, determines if they are sufficiently linearly spaced. In addition, the objects' spatial density is obtained by dividing the bounding rectangular area of the objects in a cluster by the number of objects in a cluster. These two tests provide a numeric measure of how closely the objects of a given cluster conform to the known pattern of minefield layout. The LDS algorithm is described in detail in Section 3.1.2.

3.1.1 Clustering Algorithm Analysis

The clustering algorithm is the first of two algorithms that make up the classification algorithm. A detailed description of the clustering algorithm and a determination of the processing power required is provided here. Several functions included in the source code supplied to ERIM for evaluation have been dropped from the core algorithm. The core algorithm consists of only the necessary functions of the algorithm. The omitted functions were found to be either algorithmically unnecessary artifacts of specific implementations or debugging/prototype features that would not be executed during a real-time demonstration flight. Figure 7, Core Clustering Algorithm, is an operation flow diagram of the clustering algorithm. The clustering algorithm transmits output data to the LDS process.

When first initialized, the clustering algorithm will go through a short period of higher activity during which many clusters are formed and merged. During this active period, the cluster positions will shift and thus move in IR^3 as they are merged with each other. This active period should stabilize with a set of clusters that are representative of the mines and background in the sensor terrain. Thereafter, pixels would mainly (99%) be added to the existing clusters. Occasionally, some clusters may be created. We assumed that cluster creation would stabilize since we expect the terrain to be uniform in vegetation and artifacts during the minefield search period.

In Figure 7, as each point arrives from the data compression processor, the data is converted to a cluster representation. Since a pixel can be thought of as a cluster with a single entry, we can process points and clusters with the same algorithms. The data compression processor calculates the new cluster's scaled distance to all existing clusters

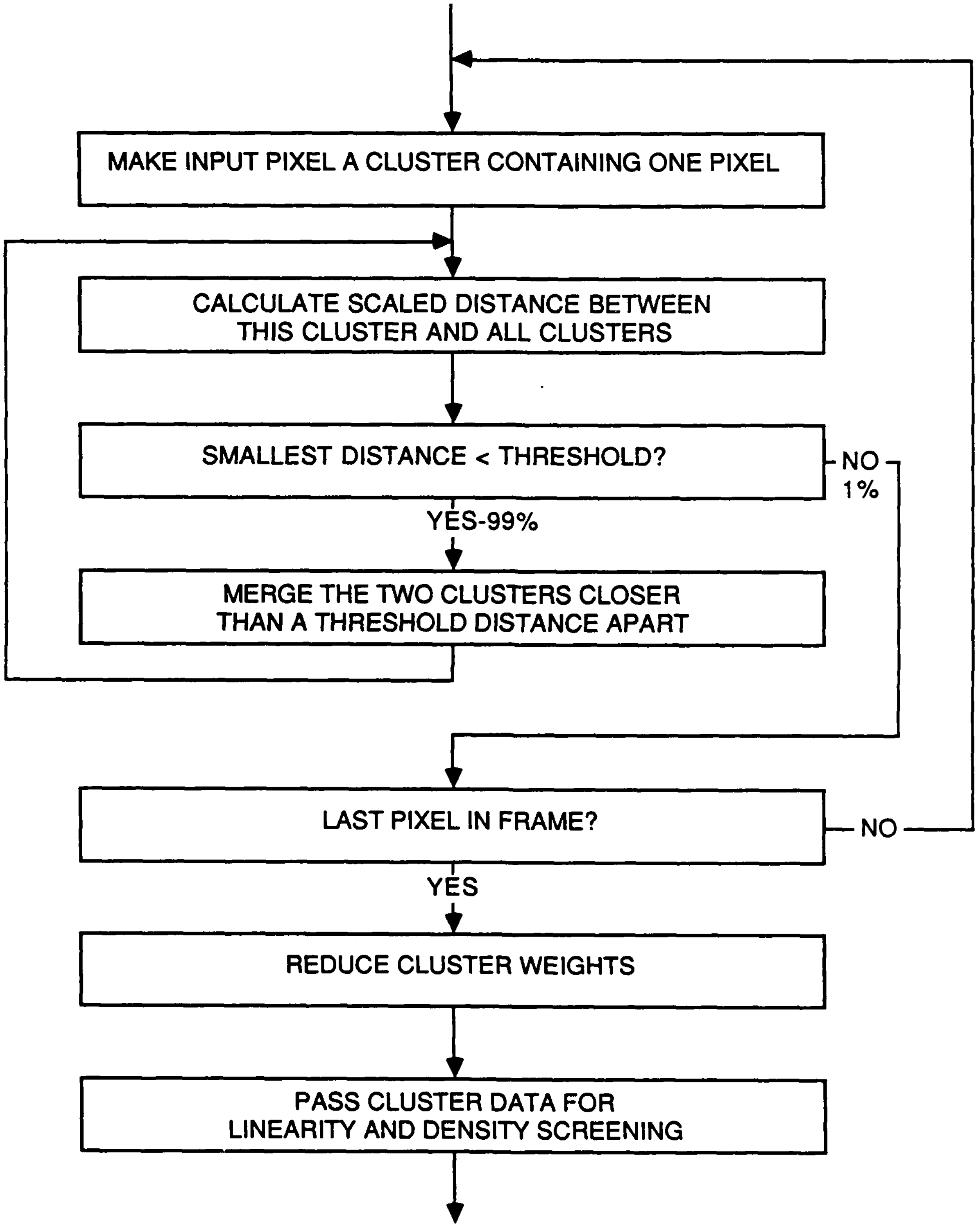
MAKE INPUT PIXEL A CLUSTER CONTAINING ONE PIXEL
CALCULATE SCALED DISTANCE BETWEEN THIS CLUSTER AND ALL CLUSTERS
SMALLEST DISTANCE < THRESHOLD?
NO 1%
YES-99%
MERGE THE TWO CLUSTERS CLOSER THAN A THRESHOLD DISTANCE APART
LAST PIXEL IN FRAME?
NO
YES
REDUCE CLUSTER WEIGHTS
PASS CLUSTER DATA FOR LINEARITY AND DENSITY SCREENING

Figure 7. Core Clustering Algorithm

and finds the smallest distance. The 99% probability is assigned to the threshold test of the minimum scaled distance because this path through the algorithm corresponds to the expected behaviour of each newly created cluster. After the cluster has been merged, the second occurence of this test will fail most of the time. This calculation of cluster distance must be done to account for any movement of the cluster that occurs when two clusters are merged. No further merging will occur (99% of the time) and the next pixel in the frame will be considered. Since each new pixel will be merged once (99% of the time), the scaled distance equation must be computed twice for each pixel in the frame.

The last two blocks of the real-time clustering algorithm in Figure 7 reduces the mass (number of points) in the clusters. The clustering algorithm executes for an extended period of time on a continuous stream of data from the IR sensor. If the external thermal conditions change and the cluster masses are not periodically reduced, the external changes could have a serious effect on the reliability of the algorithm. For instance, if the sun goes behind a cloud, the IR radiance values that formerly indicated a mine could shift to those of a benign background object. Conversely, background objects could appear to be mines.

The clustering algorithm contains several equations that make up the majority of the processing load. The evaluation of the scaled distance requires the most computation in the clustering algorithm since it requires many calculations and must be evaluated at least twice for each new pixel. Throughout the algorithm analysis, the following assumptions are made:

- Image data pixels (S_i,D_i,T_i) are 8-bit values.
- A classification frame is 700 $\times$ 512 pixels (three sequential IR sensor frames).
- A maximum of 5000 points/frame will be received from the data compression processor.
- A maximum of 50 clusters will be generated.
- The minimum object (mine) size is 5 data-compressed pixels.
- Pixel data is transferred in a raster-scan order.
- The frame rate is $\frac{1}{1.5\,seconds} = 0.66\,\text{Hz}$.

The assumptions above are consistent with the source code supplied to ERIM by WES.

We have defined several variables in order to describe the clustering algorithm mathematically. S, D, and T represent the sum and difference of the polarized sensor intensities and the thermal sensor intensity, while n_i represents the number of points in the ith cluster. The analysis of the classification algorithm's processing load assumes that $\overline{S_i}$, $\overline{D_i}$, $\overline{T_i}$, $\sum S_i$, $\sum D_i$, $\sum T_i$, and n_i are maintained for each cluster to minimize calculation of the scaled distance. $\overline{S_i}$, $\overline{D_i}$, and $\overline{T_i}$ are the averages and $\sum S_i$, $\sum D_i$, and $\sum T_i$ are the sums of all the pixels in the ith cluster. Equations (1–3) represent a form of the clustering and merging equations that requires minimal calculation.

The scaled distance is defined as:

$$\left(\frac{1}{\text{cov}}(n_j + n_i)\right)^2 \left[\left(\frac{\overline{S_j} - \overline{S_i}}{\sum S_j + \sum S_i}\right)^2 + \left(\frac{\overline{D_j} - \overline{D_i}}{\sum D_j + \sum D_i}\right)^2 + \left(\frac{\overline{T_j} - \overline{T_i}}{\sum T_j + \sum T_i}\right)^2\right] \tag{1}$$

where 'cov' is a constant. WES defines the scaled distance as the square root of Equation (1). Because the scaled distance is only used for comparison to other scaled distances or to an empirical constant, the square root is unneccesary. It also increases computation time. Optimal software coding of the scaled distance equation will require 9 additions, 3 divisions, and 6 multiplications.

When two clusters are merged, the new cluster averages $(\overline{S'_j}, \overline{D'_j}, \overline{T'_j})$ and cluster sums $(\sum S'_j, \sum D'_j, \sum T'_j)$ are recalculated from the current cluster values according to:

$$\overline{S'_j} = \frac{\sum S'_j}{(n_j + n_i)}, \qquad \overline{D'_j} = \frac{\sum D'_j}{(n_j + n_i)}, \qquad \overline{T'_j} = \frac{\sum T'_j}{(n_j + n_i)} \tag{2}$$

where:

$$\begin{aligned} \sum S'_j &= \sum S_j + \sum S_i, \\ \sum D'_j &= \sum D_j + \sum D_i, \\ \sum T'_j &= \sum T_j + \sum T_i. \end{aligned} \tag{3}$$

Optimal coding of these equations will require 4 additions and 3 divisions per merge.

The data acquisition rate allows 1.5 seconds to process 5000 points and 50 clusters. Consequently, the computational load can be computed as:

$$\begin{aligned}&\left(\frac{2(9\mathrm{A}+6\mathrm{M}+3\mathrm{D})+(4\mathrm{A}+3\mathrm{D})}{1.5\,\mathrm{seconds}}\right)\times 50\times 5000\approx \\ &(3.7\mathrm{A}+2\mathrm{M}+1.5\mathrm{D})\ \mathrm{MFLOPS}\approx 7.2\,\mathrm{MFLOPS}.\end{aligned}\tag{4}$$

The 7.2 million floating-point operations per second (MFLOPS) rate is used in Section 3.1.3 to determine the total processing load for the classification algorithm.

3.1.2 Linearity and Density Screening (LDS) Algorithm Analysis

The LDS algorithm is the second of two algorithms that make up the classification algorithm. A detailed description of the LDS algorithm and a determination of the processing power required is provided here. The LDS algorithm groups pixels of the same cluster into physical objects based on their spatial (x, y) locations within the original IR images. All the objects within a cluster are subjected to a linear regression test to determine the linearity of the objects in the original IR images. The spatial densities of the objects of each cluster are then calculated. The linearity and density values together provide a quantitative test of whether a given cluster of objects is a minefield.

Figure 8, Linearity and Density Screening Algorithm, is an operation flow diagram of the LDS algorithm. The LDS algorithm accumulates a frame of up to 5000 points that were selected by the data compression processor. It then uses the points' cluster identification (a value that uniquely identifies the cluster that the point is a member of) and (x, y) image coordinates to group the points into objects with the same cluster identification and similar location. The objects in each cluster are then analyzed to determine their linearity with respect to one another and the density of the cluster. Finally, the LDS algorithm selects the clusters with acceptable linearity and density values and ranks them according to their likelihood of being minefields.

The processing load for the LDS algorithm can be determined in a manner similar to that of the clustering algorithm. Each section in Figure 8 has a separate processing load that must

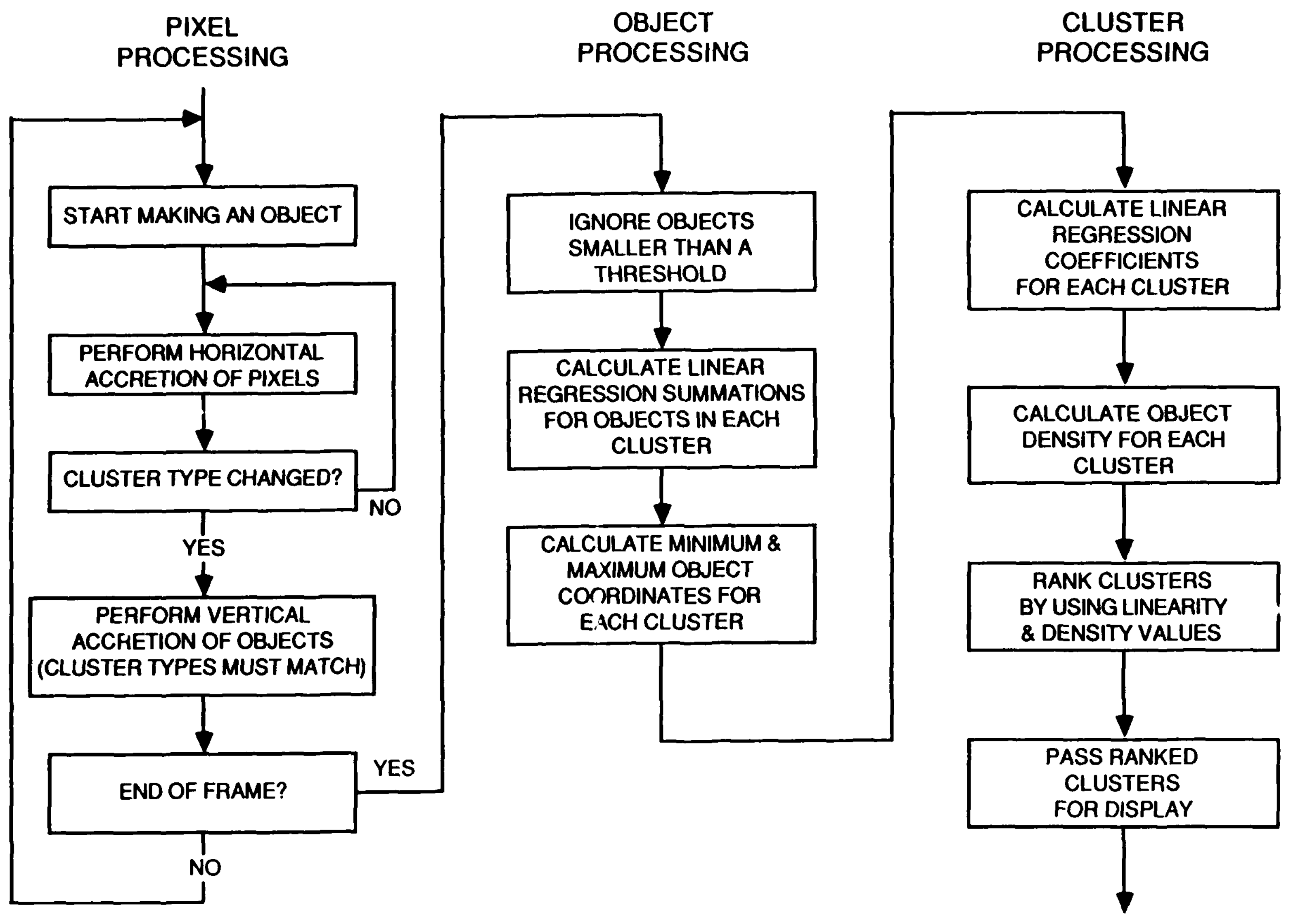
PIXEL PROCESSING
START MAKING AN OBJECT
PERFORM HORIZONTAL ACCRETION OF PIXELS
CLUSTER TYPE CHANGED?
NO
YES
PERFORM VERTICAL ACCRETION OF OBJECTS (CLUSTER TYPES MUST MATCH)
END OF FRAME?
YES
NO
OBJECT PROCESSING
IGNORE OBJECTS SMALLER THAN A THRESHOLD
CALCULATE LINEAR REGRESSION SUMMATIONS FOR OBJECTS IN EACH CLUSTER
CALCULATE MINIMUM & MAXIMUM OBJECT COORDINATES FOR EACH CLUSTER
CLUSTER PROCESSING
CALCULATE LINEAR REGRESSION COEFFICIENTS FOR EACH CLUSTER
CALCULATE OBJECT DENSITY FOR EACH CLUSTER
RANK CLUSTERS BY USING LINEARITY & DENSITY VALUES
PASS RANKED CLUSTERS FOR DISPLAY

Figure 8. Linearity and Density Screening Algorithm

be summed to obtain the total LDS processing load. Conditional execution statements (if-then-else) are assigned a probability of execution for each path. The probability corresponds to the maximum computational load value (consistent with the other assumed constraints of the problem) that was itemized in Section 3.1.1.

The first column builds objects from pixels by first collecting pixels in a horizontal direction that are within a specified distance from each other. If the cluster identification changed then the current object boundary has been reached and a new object is started. When this occurs the object is combined in a similar manner in the vertical direction, using previously formed objects that have the same cluster identification.

Both the x and y components of point locations, which have the same class, are summed to obtain the x and y arithmetic means of the object. The means will be used as the center of mass coordinates of the objects when linearity measures are performed. The minimum and maximum x are maintained for the object and are used to vertically merge the current object segment to previous segments with the same class.

A total of 5 additions are required to test if a new point is part of the current object. Five additions are required to add a point to an object and this occurs with a probability of 0.5. To search vertically requires 90 additions; the probability of a merge is 0.5, assuming 20 previous objects are searched and 10 objects are in the same cluster as the current object. These assumptions follow from the uniform distribution of the mines and the minimum mine size of five data compressed points. Merging object sections requires 5 additions at a probability of 0.25 while starting a new object requires no arithmetic operations and has the same probability. The total number of additions is:

$$(5A) + .5(5A) + .5(90A + .5(5A)) = 53.75\,\text{additions}.$$

This value is a per point average load based on the given data-dependent execution probabilities.

The second column calculates linear regression summations for all the objects in each cluster and also finds the minimum and maximum (x, y) extent of the potential minefield boundary for each class. The variables $\mathrm{x_c}$ and $\mathrm{y_c}$ represent the center of mass coordinates of objects in the image. The regression summations must be computed for $\mathrm{x_c}$, $\mathrm{y_c}$, $\mathrm{x_c y_c}$, $\mathrm{x_c^2}$, $\mathrm{y_c^2}$, and n_i, where n_i is the number of objects in the ith class. Because the minimum object size is 5 pixels, a maximum of $\frac{5000}{5} = 1000$ objects could be in a frame. With optimal coding, 20 additions, 13 multiplications, and 2 divisions are required per object in the frame to calculate these summations.

The third column calculates the linear regression coefficients for the ith cluster:

$$\mathrm{b_1} = \frac{\sum \mathrm{x_c y_c} - \bar{\mathrm{x}}_c \sum \mathrm{y_c}}{\sum \mathrm{x_c^2} - \bar{\mathrm{x}}_c \sum \mathrm{x_c}} \qquad \text{and} \qquad \mathrm{b_0} = \bar{\mathrm{y}}_c - \mathrm{b_1} \bar{\mathrm{x}}_c,$$

where:

$$\bar{\mathrm{x}}_c = \frac{\sum \mathrm{x_c}}{\mathrm{n}_i} \qquad \text{and} \qquad \bar{\mathrm{y}}_c = \frac{\sum \mathrm{y_c}}{\mathrm{n}_i}.$$

The length of the minefield along its long axis—a complex conditional equation that depends on the orientation of the minefield in the image—is also calculated. Assuming a random orientation of the minefield implies a 0.5 probability for all the conditional pathways in calculating the minefield front length. These equations require an average of 13 additions, 7 multiplications, and 4 divisions per cluster.

The total LDS algorithm processing rate is:

$$\begin{aligned} &\frac{5000(53.75\mathrm{A}) + 1000(20\mathrm{A} + 13\mathrm{M} + 2\mathrm{D}) + 50(13\mathrm{A} + 7\mathrm{M} + 4\mathrm{D})}{1.5} \\ &\approx (0.29\mathrm{A} + 0.01\mathrm{M} + 0.0\mathrm{D})\,\mathrm{MFLOPS} \approx 0.3\,\mathrm{MFLOPS} \end{aligned} \tag{5}$$

for continuous real-time performance. This is a fairly trivial rate compared to the clustering algorithm, provided that the classification processor does not impose a severe overhead to perform the necessary operations.

3.1.3 Classification Processing Load

A realistic determination of the processing load for the entire classification algorithm is provided here. The value is the sum of the arithmetic loads determined in Sections 3.1.1

and 3.1.2 plus external communication overheads. The processing load value must include sufficient head room so that unforeseen complexities in the implementation of the algorithms will not require the selected classification processor to be replaced by one with more processing power.

The clustering and LDS algorithms require a minimum numeric processing load of about 7.5MFLOPS consisting of 4.0M additions, 2.0M multiplications, and 1.5M divisions. The 7.5MFLOPS value must realistically be doubled to allow for non-optimal coding and unforeseen problems in implementing the algorithms for real-time performance. Doubling the required load is a rough rule of thumb in accordance with good engineering practice. Therefore, the required numeric processing rate for the clustering and LDS algorithms, not considering other overhead sources, increases to 15MFLOPS.

Additional processor time will be required for the overhead instructions of program flow control, data loading, etc. Numerous instruction traces for vonNeumann processors put these types of overhead at less than 20 percent of the total instruction stream for this kind of a numerically intensive algorithm. In addition, these operations are mainly of simple integer type and execute much faster than the more complex floating-point arithmetic instructions. Consequently, an additional load of about 3 million instructions per second (MIPS) should be added to the total processor load.

Communication with the data compression processor is an additional source of processing load. The communication involves receiving the selected pixels with their (x, y) coordinates and transmitting back the locations and rankings of individual mines after classification. The input pixels and locations will likely consist of one 32-bit word containing the three IR values (S_i, D_i, T_i) and another 32-bit word containing the associated (x, y) coordinates (16 bits each). Therefore, the data transfer rate is 40,000 bytes (5000×8) in 1.5 seconds or about $26,700$ bytes per second. Sending the classification results back for display would require the transfer of a maximum of 1000 objects, consisting of 5 bytes each (for x, y, and rank), in 1.5 seconds. This is about $3,300$ bytes per second. The total is 30,000 bytes per second. Although the communication rate appears small, specific architectures may have large I/O overheads. If ten instructions are required per byte moved, then an additional 0.3MIPS should be added into the classification algorithm processing load.

In summary, the classification algorithm will require about 15MFLOPS and 3–4MIPS to maintain real-time execution plus 0.3MIPS for data compression to classification processor communication. Any additional overhead that a given processor's architecture imposes on the algorithm will be discussed in Sections 3.2.1 through 3.2.5.

3.1.4 Fixed-Point Precision Requirements

An estimate of the precision requirements of classification Equations (1-3) is provided here. The required precision is used to determine the minimal precision (number of bits) that will provide accurate classification results. The precision required to evaluate Equations (1–3) is estimated by considering the range of the quantities they contain and the operations with which they are used.

The following assumptions are used in the analysis:

- The 'cov' is an 8-bit quantity.
- Equation (1) is required to have 8-bit accuracy.
- The maximum number of points in the ith cluster is $n_i = 255$.

Section 3.1.1 states that $\overline{S_i}$, $\overline{D_i}$, $\overline{T_i}$, $\sum S_i$, $\sum D_i$, $\sum T_i$, and n_i are required to compute the distance and merging equations for each cluster. The number of points (n_i) in the ith cluster requires a minimum of 8 bits ($\text{Log}_2 255$) of precision. $\sum S_i$, $\sum D_i$, and $\sum T_i$ each have a range from 0 to 255^2 (255 points $\times$ 255 intensities) and require 16 bits ($\text{Log}_2 65025$) of precision each. $\overline{S_i}$, $\overline{D_i}$, and $\overline{T_i}$ have the same range as S_i, D_i, and T_i (0–255) because they each contain n pixels and are each divided by n. Thus, their precisions will also be 8 bits. The precisions of the individual values can be used along with Eqvations (1–3) to calculate the maximum precision necessary for the clustering algorithm.

Equation (1) of Section 3.1.1 can be divided into several parts.

$$\left(\frac{1}{\text{cov}}(n_j + n_i)\right)^2 \tag{6}$$

requires:

$$2\left[\mathrm{Log}_2\left(\frac{1}{1}(255+255)\right)\right] \approx 18 \text{ bits} \tag{7}$$

of precision to the left of the decimal point for the worst-case values (cov = 1, $n_i = n_j = 255$) and:

$$2\left[\mathrm{Log}_2\left(\frac{1}{255}(1+1)\right)\right] \approx 14 \text{ bits} \tag{8}$$

of precision to the right of the decimal point for the worst-case values (cov = 255, $n_i = n_j = 1$). The denominators in the three terms of Equation (1):

$$\left(\frac{\overline{S_j}-\overline{S_i}}{\sum S_j + \sum S_i}\right)^2, \quad \left(\frac{\overline{D_j}-\overline{D_i}}{\sum D_j + \sum D_i}\right)^2, \quad \left(\frac{\overline{T_j}-\overline{T_i}}{\sum T_j + \sum T_i}\right)^2, \tag{9}$$

are about 2n times as large as each numerator because the denominator contains twice the sum of n values that have the same range as those in the numerator. The required precision is:

$$\mathrm{Log}_2((2n)^2) = 18 \text{ bits} \tag{10}$$

to the right of the decimal point when the worst-case (n = 255) is used.

To compute the precision to the left of the decimal point for these terms, we must first consider the order that the terms are evaluated. Optimal coding for speed requires the squaring to occur last so that only one squaring operation is performed for each term. Previous to squaring, the numerator and denominator are evaluated and their quotient is formed. The numerator will never be larger than the range of $\overline{S_i}$, $\overline{D_i}$, and $\overline{T_i}$ (255 with a precision of $\mathrm{Log}_2 255 = 8$ bits) and the denominator never larger than twice the range of $\sum S_i$, $\sum D_i$, and $\sum T_i$ ($2 \times 65025 = 130050$ with a precision of $\mathrm{Log}_2 130050 = 17$ bits). Division of the numerator by the denominator will place the precision to the right of the decimal point. Consequently, the minimal precision required to the left of the decimal point is 17 bits.

Equations (1–10) provide enough information for an estimation of the overall fixed-point precision requirements of the clustering and merging algorithms. Eighteen bits are necessary to the left of the decimal point to represent the maximum value of Factor (6). Eighteen bits are necessary to the right of the decimal point to retain sufficient accuracy when the fractional Terms (9) are evaluated. The total requirement is 36 bits.

In conclusion, 32-bit floating-point representation will best support the accuracy and range of values required for the clustering and merging algorithms. The accuracy required at each step in the evaluation of Equations (1–3) never exceeds the 24-bit precision of the mantissa of a 32-bit floating-point value. From a resource point of view, the 36 bits could be used more efficiently—provide a larger range of the same accuracy using the same precision—if they were used to represent floating-point variables.

The classification algorithm has been defined and its required precision determined; we can use this information in the classification hardware performance analysis below.

3.2 Classification Hardware Performance Analysis

Five architectures were considered for the implementation of the classification algorithms. Two fine-grained, single instruction multiple data (SIMD) architectures were evaluated. The SIMD architectures are made up of many—1024 in both of these processors—simple parallel PEs, configured with linear (AISI5000) and mesh (DAP500) communication paths. The analysis for the AISI AISI5000 and the AMT DAP500 architectures are contained in Section 3.2.5 and Section 3.2.1, respectively. Three different coarse grained multiple instruction multiple data (MIMD) architectures were evaluated: one using standard complex instruction set computer (CISC) microprocessors communicating on a parallel bus interconnection topology (68020 microprocessor), one using custom CISC microprocessors communicating on a hypercube interconnection topology (N-Cube 7), and one using reduced instruction set computer (RISC) processors communicating with a flexible mesh interconnection topology (T800 Transputer). The analyses for the Force 68020, N-Cube N-Cube 7, and Parsytec T800 Transputer are contained in Section 3.2.4, Section 3.2.3, and Section 3.2.2, respectively.

3.2.1 AMT DAP 500

One of the two implementations of an SIMD, mesh connected architecture—AMT's (Active Memory Technology) DAP 500 (Distributed Array Processor)—is discussed here. The DAP 500 is organized as a 32×32 mesh of single-bit PEs and up to 1Mbit of local memory per element (see Figure 9, DAP 500 System and Figure 10, DAP 500 Processor Element Organization). The system uses a 10 MHz clock rate. Most instructions can be executed at this rate to provide 10^{10} Boolean operations per second. These features make the DAP 500 well suited for vector, matrix, and image operations.

The analyses contained in the fixed-point and floating-point sections below use the classification algorithm MFLOPS rate, which was calculated in Section 3.1, and the fixed-point and floating-point processing rates of the DAP 500, along with some approximations for the amount of processing power consumed by overhead items, to determine how much head room the DAP 500 provides. If the head room—defined as the amount of extra processing power available in the basic system—approaches 100%, then the DAP 500 passes the mandatory criteria for processing power.

3.2.1.1 Fixed-Point Performance Analysis

We can use the fixed-point precision requirements to calculate the amount of time the DAP 500 will require to execute the clustering and LDS algorithms. First the execution time for processing Equation (1) is calculated using execution times published by AMT for 1024-element vector operations using 32-bit fixed-point representation. The analysis is then repeated using actual times calculated by AMT for Equation (1), assuming 1024-element vector operation, using 16-bit fixed-point representation. Results for the total clustering and LDS algorithm execution time are extrapolated from the execution time of Equation (1) and normalized for comparison. By comparing these times with the 1.5 second frame time, we can calculate the head room provided by the DAP 500.

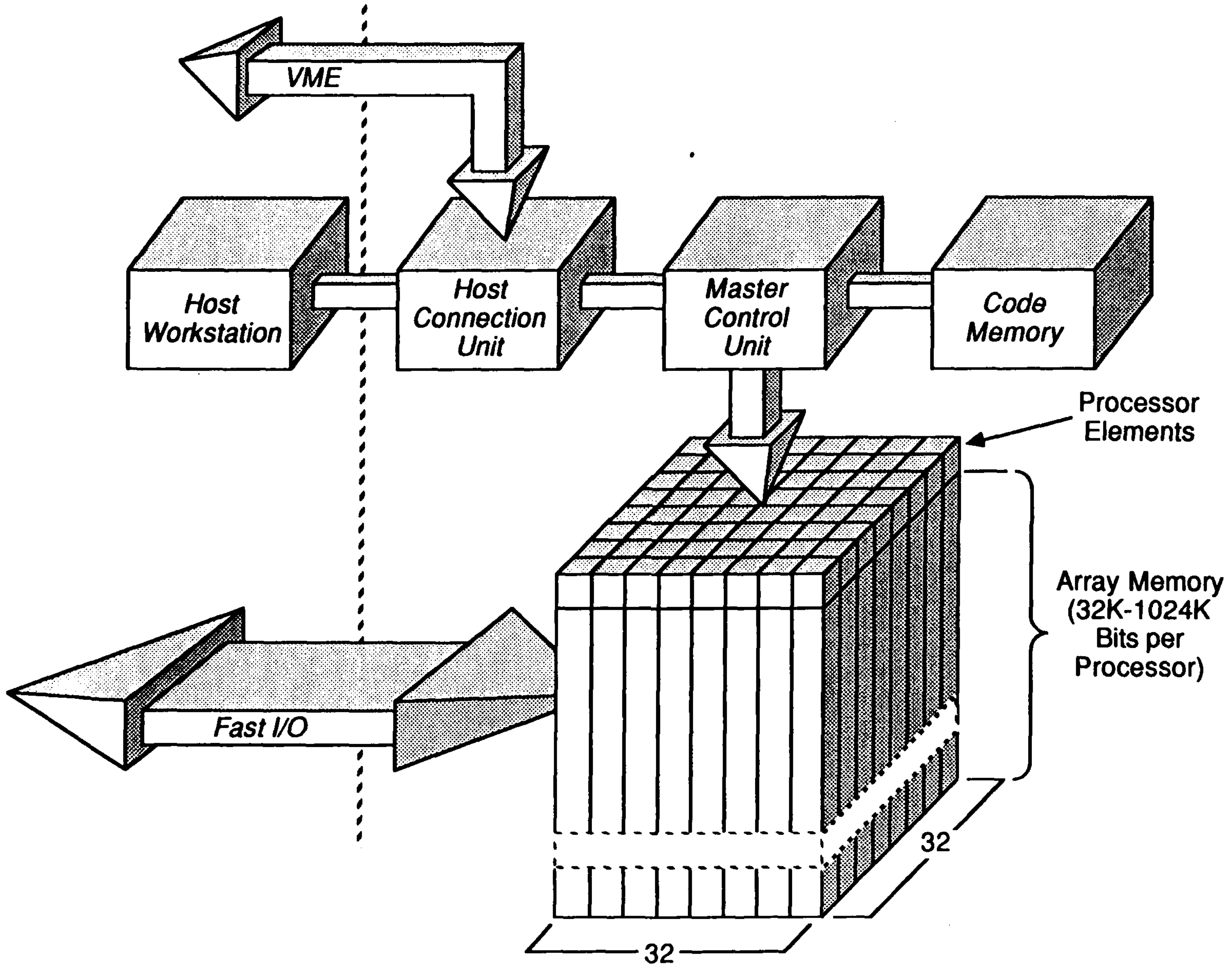

Figure 9. DAP500 System

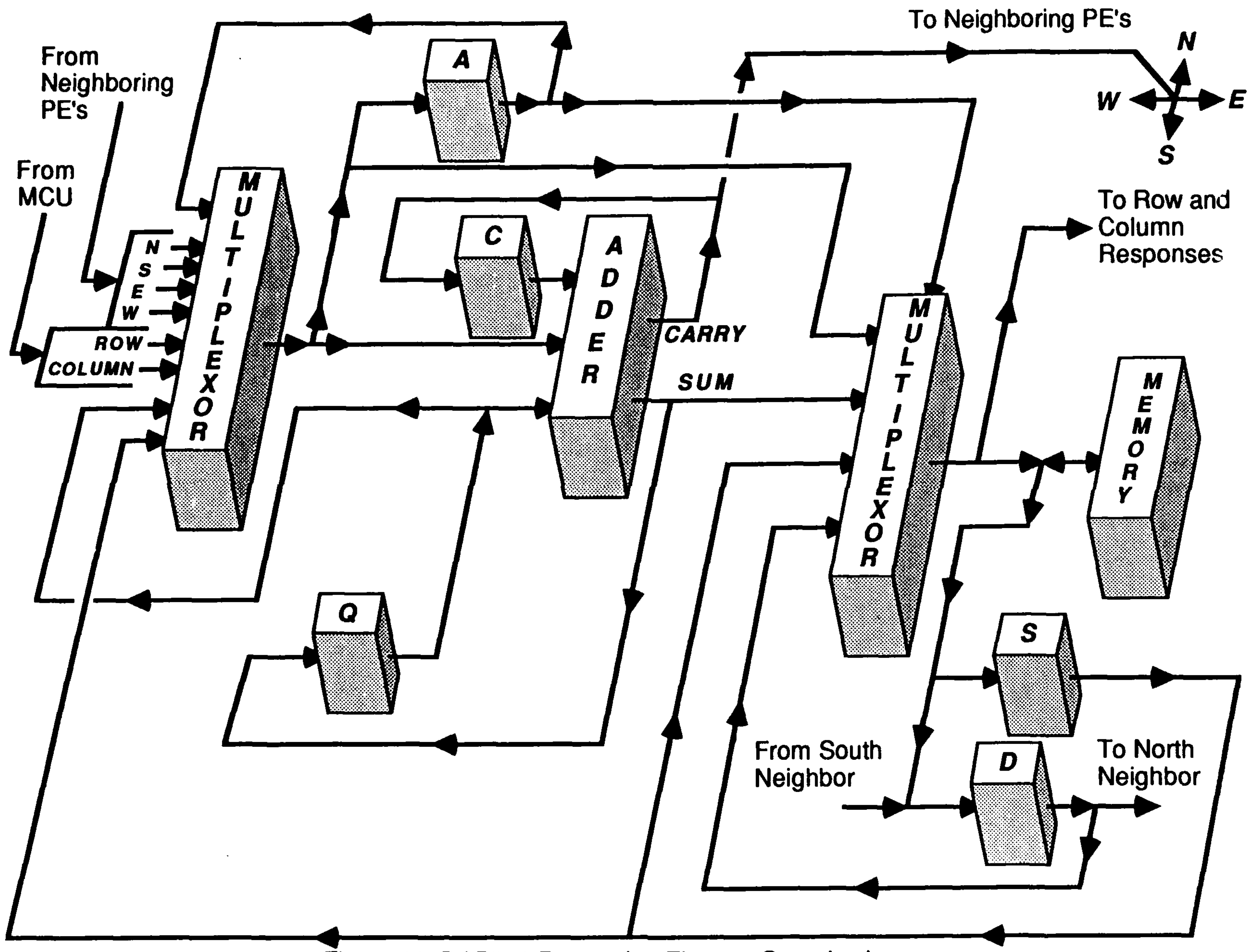

Figure 10. DAP500 Processing Element Organization

The DAP500 is capable of executing:

- 1024 32-bit fixed-point additions or subtractions in 12μsec or
- 1024 32-bit fixed-point multiplications or divisions in 190μsec.

When using additional bits of precision, we assume a linear increase in time for addition and subtraction operations and a geometric increase of time for multiplication and division operations. Consequently, we find that the DAP500 is capable of executing:

- 1024 36-bit fixed-point additions or subtractions in 13.5μsec or
- 1024 36-bit fixed-point multiplications or divisions in 240.47μsec.

The time to calculate the clustering and merging equations of 5000 points for 50 clusters is then:

$$50 \times 5000 \times \left(22 \frac{13.5\mu\text{sec}}{1024} + 21 \frac{240.47\mu\text{sec}}{1024}\right) = 1.31\,\text{seconds.}$$

The LDS calculations were estimated to require 0.3MFLOPS; this is 5% of the clustering and merging processing load. AMT estimates that data I/O consumes about 8% of the processing power of the DAP500. Therefore, we increase the time by 5% to account for the LDS calculations:

$$1.31\,\text{seconds} + 0.05 \times 1.31 = 1.37\,\text{seconds}$$

and then increase it by 8% to account for data I/O:

$$1.37\,\text{seconds} + 0.08 \times 1.37 = 1.48\,\text{seconds.}$$

The 1.48 seconds is very close to the allotted processing time of 1.5 seconds.

In conclusion, 36-bit fixed-point processing will just barely allow the processing to be done at real-time rates. The head room is:

$$\frac{(1.50 - 1.48)}{1.48} \times 100\% = 1\%$$

This pecentage of head room does not approach the 100% head room that was mandated by good engineering practice.

Actual processing times for the scaled-distance equation, shown in Equation (1) and evaluated at AMT, are used to provide a more realistic value of the DAP500 head room. The following additional assumptions were used:

- 16-bit fixed-point format is appropriate for the problem,
- All 1024 calculations would have the same fixed-point representation and would require identical normalization, and
- No processors would be idle in any given frame.

The measured running time was 735μsec for 1024 evaluations, which is equivalent to:

$$\frac{50 \times 5000}{1024} 735\mu\text{seconds} = 0.18\,\text{seconds}$$

for the 250,000 (50 × 5000) evaluations required in the worst-case scenario.

These calculations were performed using 16-bit precision and must be normalized with respect to the 36-bit fixed-point processing times computed previously. Linear increases for addition and subtraction and geometric increases for multiplication and division will be used again. Section 3.1.1 shows that the distance equation consists of 9 additions and subtractions and 9 multiplications and divisions. The time to compute 250,000 evaluations of Equation (1) is:

$$\frac{36}{16}\left(\frac{9}{18}(0.18)\right) + \left(\frac{36}{16}\right)^2 \left(\frac{9}{18}(0.18)\right) = 0.66\,\text{seconds.}$$

The total running time for the clustering algorithm requires a consideration of the following. Equation (1) must be computed at least twice per frame. The merging equations must also be computed once per frame for each of the 5000 points, times each of the 50 clusters. Because Equation (1) requires 9 additions, 3 divisions, and 6 multiplications and Equation (2) requires 4 additions and 3 divisions, the approximation of the running time for the merging equations is less than 50% of the scaled distance running time:

$$\left(\frac{4+3}{9+3+6}\right) 0.66 \approx 0.3\,\text{seconds.}$$

The total running time for the evaluations of the clustering equation is:

$$0.66 + 0.66 + 0.3 = 1.61\,\text{seconds.}$$

The LDS calculations were estimated to require 0.3 MFLOPS; this is 5% of the clustering and merging processing load. AMT estimates that data I/O consumes about 8% of the processing power of the DAP 500.

Therefore, the time can be increased by 5% to account for the LDS calculations:

$$1.61\,\text{seconds} + 0.05 \times 1.61 = 1.69\,\text{seconds}$$

and then increased by 8% to account for data I/O:

$$1.69\,\text{seconds} + 0.08 \times 1.69 = 1.83\,\text{seconds}.$$

This time is a reasonable estimate of the entire processing time per frame including data I/O. Comparing 1.83 seconds with the 1.5 seconds available for these calculations shows that there is no head room available for overhead items that haven't been included in this analysis or for an increase in the required processing power of the classification algorithm.

There are a few factors that haven't been included in calculating the 1.83 second processing time. The classification calculations could be hand coded by an experienced DAP 500 programmer to make use of the full power of the system. The algorithm complexity has been estimated, so more processing power may be required. There are other overhead items such as interprocessor communication and synchronization that may also require a significant percentage of the total processing power available. The maximum number of clusters is not related to the processor mesh size (32×32) and may require that a percentage of the processors are unused for a portion of each iteration.

3.2.1.2 Floating-Point Performance Analysis

The amount of time that the DAP 500 requires to execute the classification and LDS algorithm equations, when a floating-point representation for variables is used, is estimated here. This time is compared to the 1.5 second frame time limit for real-time processing to determine if the required head room is available.

The DAP 500 is capable of executing:

- 1024 32-bit floating-point additions or subtractions in 105μsec; this is 9.75 MFLOPS.
- 1024 32-bit floating-point multiplications in 156μsec; this is 6.56 MFLOPS.
- 1024 32-bit floating-point divisions in 212μsec; this is 4.83 MFLOPS.

If the algorithm exclusively contained multiplications or divisions, the DAP 500 would not be able to provide 7.5 MFLOPS for the real-time classification algorithm. Therefore, a closer examination of the algorithm is necessary.

Referring to Equations (1–3), we see that there are $9+9+4 = 22$ additions, $3+3+3 = 9$ divisions, and $6+6+0 = 12$ multiplications required to calculate the clustering and merging equations for a single point on one cluster. The time to calculate the clustering and merging equations of 5000 points for 50 clusters is:

$$50 \times 5000 \times \left(22\frac{105\mu\text{sec}}{1024} + 9\frac{156\mu\text{sec}}{1024} + 12\frac{212\mu\text{sec}}{1024}\right) = 1.53\,\text{seconds.}$$

The LDS calculations will also require a relatively small amount of time . ıce they were estimated to require 0.3 MFLOPS; this is 5% of the clustering and merging processing load. Also, AMT estimates that data I/O consumes about 8% of the processing power of the DAP 500.

Therefore, the time can be increased by 5% to account for the LDS calculations:

$$1.53\,\text{seconds} + 0.05 \times 1.53 = 1.61\,\text{seconds}$$

and then increased by 8% to account for data I/O:

$$1.61\,\text{seconds} + 0.08 \times 1.61 = 1.74\,\text{seconds.}$$

This time is a reasonable estimate of the entire processing time per frame including data I/O.

Comparing this time with the 1.5 seconds available for these calculations shows that there is no head room available for overhead items that haven't been included in this analysis or for an increase in the required processing power of the classification algorithm.

As in section 3.2.1.1, there are a few factors that haven't been considered. The classification calculations could be hand coded by an experienced DAP500 programmer to make use of the full power of the system. The algorithm complexity has been estimated, so more processing power may be required. There are other overhead items such as interprocessor communication and synchronization that may also require a significant percentage of the total processing power available. The maximum number of clusters is not related to the processor mesh size (32×32) and may require that a percentage of the processors are unused for a portion of each iteration.

Based on these head room values and those of the fixed-point performance analysis, another processor should be considered for implementing the classification algorithm.

3.2.2 Parsytec T800 Transputer

An evaluation of a RISC processor communicating by means of a flexible mesh interconnection topology is discussed here. A multiple transputer system from Parsytec GmbH can be configured to perform all the processing functions necessary in the classification processor. Figure 11, Transputer-based Classification System, illustrates the system level interconnection that would be used to complete the two separate algorithms. The tree structure on the left implements the clustering algorithm, while the two processors on the right alternately perform the LDS algorithm on a frame-by-frame basis. When processing at real-time rates, pixel data enters the clustering tree and classified pixels flow out to the LDS processors. A full frame of pixels are accumulated alternately by each processor.

Since the clustering algorithm requires a high processing rate to maintain real-time execution rates, it must be shared by several Transputers. The binary tree structure chosen facilitates the required search for the smallest scaled distance within the clustering algorithm. This structure also minimizes the time required to broadcast the value of a merged cluster or a new pixel entering the algorithm during the algorithm's iterative step.

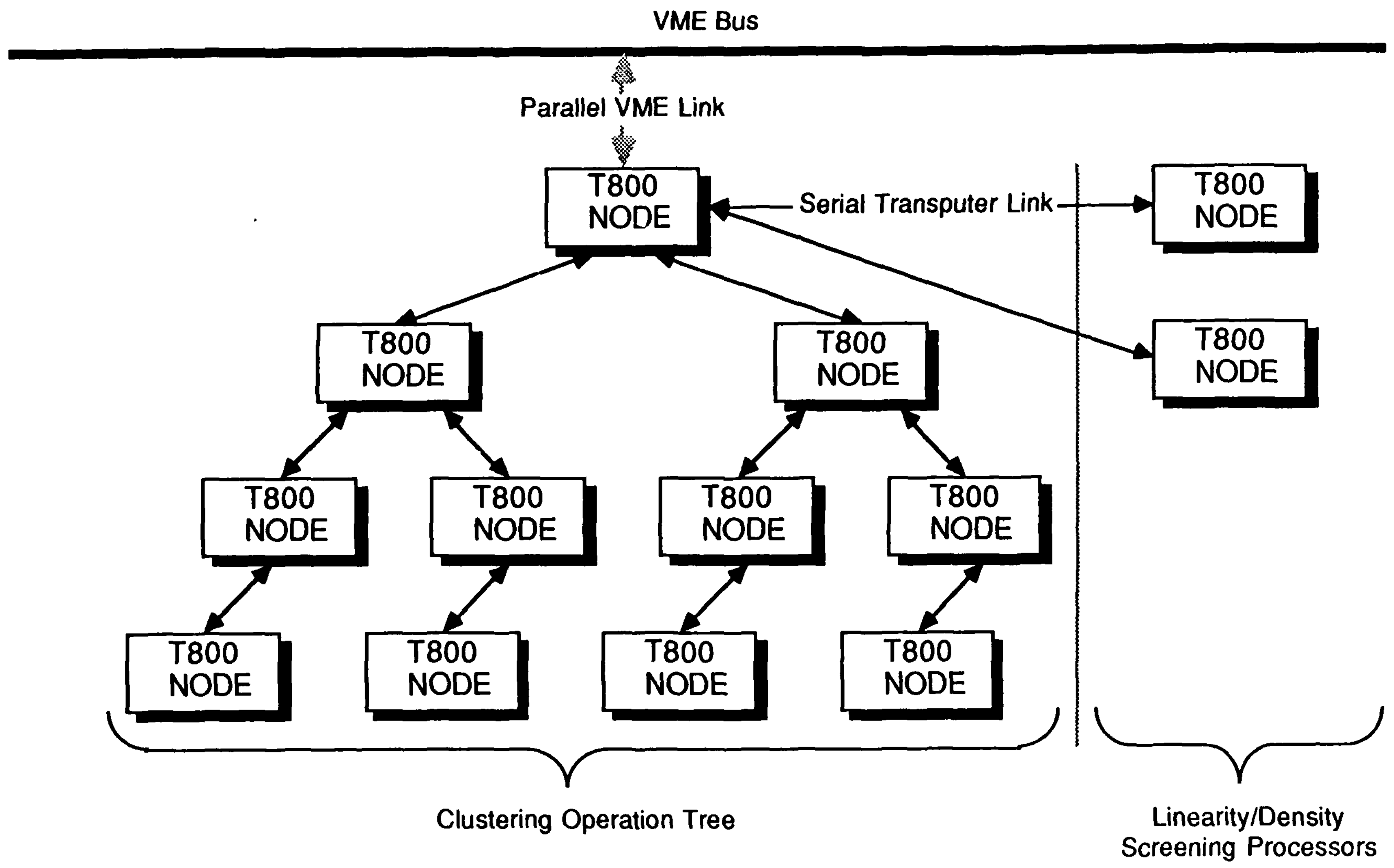
VME Bus
Parallel VME Link
Serial Transputer Link
T800 NODE
Clustering Operation Tree
Linearity/Density Screening Processors

Figure 11. Transputer-based Classification System

The LDS algorithm requires a small processor load that could be provided by a single transputer. Two transputers are used because LDS processing changes character from pixel-by-pixel to frame-by-frame. A smooth transition from one type of processing to the other is provided by having one transputer process a frame of pixel data while the other completes the calculations that require a complete frame—up to 5000 pixels—of data to accumulate before processing can proceed. The results are then passed along to the data compression processor for display.

Central control of the multi-processor is executed by the first processor in the clustering tree. The processor communicates with the data compression processor over the industry standard VME bus. The primary messages passed are the initially selected pixels output by the data compression processor and the final mine locations passed back from the classification processor.

3.2.2.1 T800 Processor Capabilities

The transputer nodes shown in Figure 11 consist of 1 MByte of local memory, a T800 transputer that includes four standard transputer links for communication, 4 kBytes (4×2^{10} Bytes) of fast RAM, and an Institute of Electrical and Electronics Engineers (IEEE) floating-point processor. Figure 12, T800 Transputer Chip Organization, illustrates the internal architecture of the T800 processor module and the internal resource interconnections. The integer and floating-point units can maintain a throughput of 7 MIPS and 2 MFLOPS—using 32-bit representation—respectively, when data resides in the on-chip memory. In addition, the T800 control logic allows floating-point, integer, and communications operations to be overlapped; the T800 is limited by access to the internal memory and interdependencies between instruction operands.

The transputer links are a primary attraction of these processors as they provide a low overhead communication channel between programs running on two different processors. The transputer system program is initially loaded over the links on power-up from a non-volatile memory on the VME bus. During real-time execution, all data I/O occurs over the

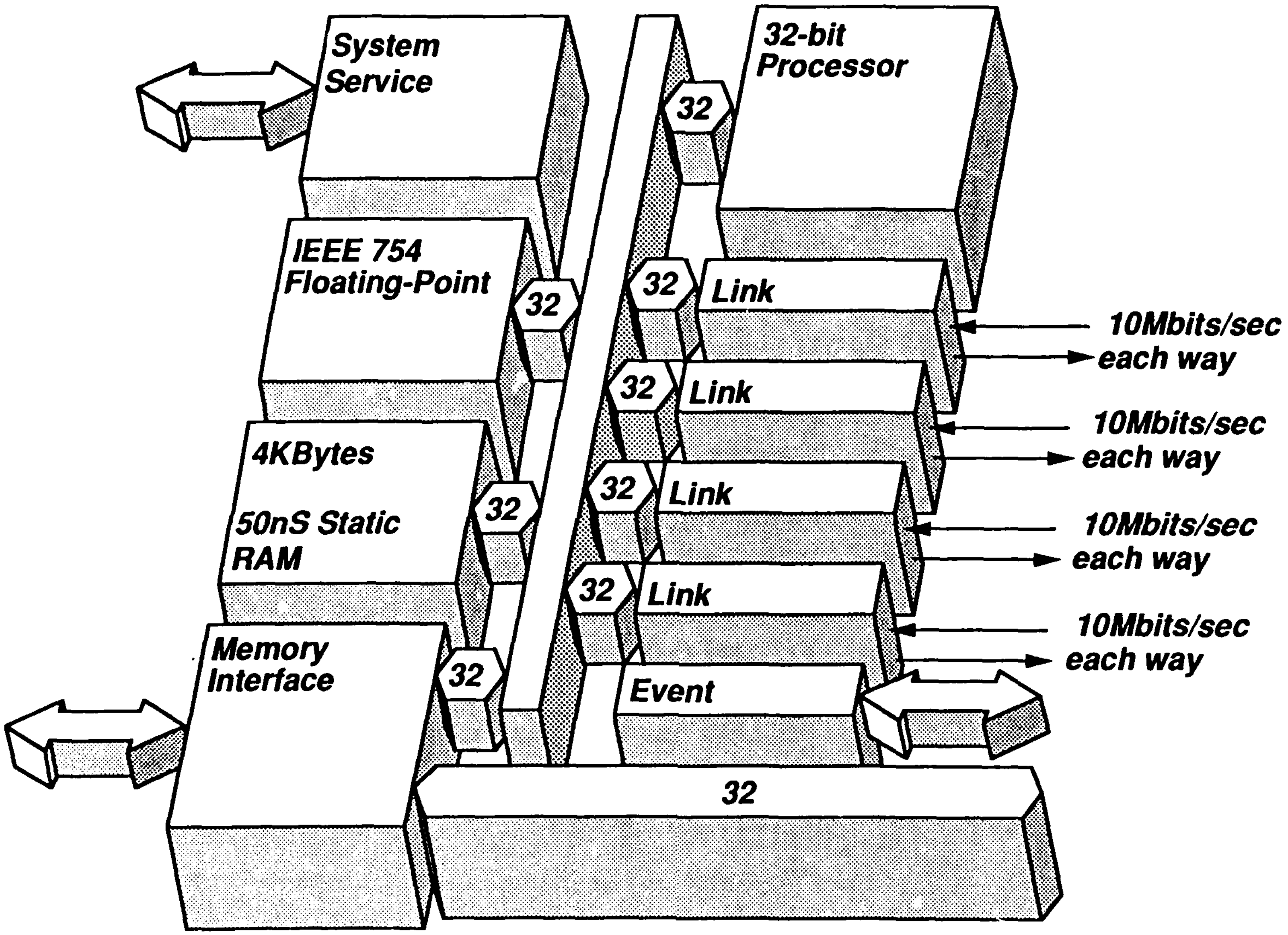

Figure 12. T800 Transputer Chip Organization

links. The eight byte messages required for the classification algorithm can be transmitted at 10 μsec per message.

3.2.2.2 T800 Classification Performance

The T800 processor discussion shows that a network of transputers have sufficient power to execute the classification algorithms. Whether the algorithm should be implemented in floating-point or fixed-point is a significant question. A floating-point algorithm is much simpler to code and debug because fixed-point range overflows cannot occur. However, floating-point operations are generally more complex and on some machines much slower than their fixed-point counterparts.

For single precision IEEE floating-point, the T800 can execute single additions, multiplications, and divisions in 0.35 μsec, 0.55 μsec, and 0.85 μsec, respectively.

Based on the ratios of operations contained in the clustering and merging Equations (4–5), an aggregate rate of:

$$\frac{8(0.35)+4(0.55)+3(0.85)}{15} = 0.503\,\mu\text{sec} \quad \text{or} \quad \approx 2.0\,\text{MFLOPS}$$

is required, assuming 32-bit representation.

Using 32-bit fixed-point, the transputer can perform single additions, multiplications, and divisions in 0.05μsec, 1.95μsec, and 2.0μsec respectively. These times correspond to a throughput rate of 1.06 MIPS for the given instruction mix or about half as fast as the floating-point version. The floating-point unit is faster than the integer unit on the T800 because it was designed several years after the integer unit and uses newer technology.

In addition to the arithmetic operations, several communication operations occur during the calculations for each input point of the clustering algorithm. Input pixels are broadcast to all the processors for calculation of the scaled distance. Then each processor finds the minimum distance and sends it toward the root of the tree where the root node checks the distance against a threshold. Either a merge command is sent back or the pixel remains a discrete cluster. In all, a worst-case value of about 100 bytes must be shipped around the multiple processors at 1.25μsec per byte, or 125μsec per input pixel. Much of the data transfer time is overlapped with the arithmetic processing, but a 20% reserve capacity in the clustering algorithm will be required to ensure that real-time processing rates can be maintained when communication overhead is included.

The preceding throughput calculations show that nine T800s are required for the clustering algorithm (9×2.0 MFLOPS $>$ 1.2(14.4) MFLOPS) and the required 20% reserve for internal communication. As already stated, two additional T800s would be used for the LDS. The Parsytec transputers are most economically purchased in multiples of two, so a total of twelve T800s would be used, with ten assigned to the clustering algorithm. In addition, central control would be accomplished by interfacing the twelfth T800 transputer with the VME bus. The system described would have more than the desired 100% performance margin and would be economically priced due to the inherent simplicity of transputer-based computers.

3.2.3 N-Cube N-Cube7

An evaluation of a group of CISC microprocessors communicating on a hypercube interconnection topology is discussed here. The N-Cube7, from N-Cube Systems, is a MIMD processor organized as a hypercube connection of several 0.5MFLOPS CISC node processors, each with 128kBytes of local memory (see Figure 13, Complete N-Cube10 System, and Figure 14, N-Cube Processor Chip Organization). Each node has direct connections to N nodes where N represents the dimension of the N-Cube and can communicate to every node in the hypercube by using neighboring nodes as switching networks. Any two nodes can communicate using less than N nodes for communications routing, not counting the source and destination nodes, in an N-dimensional hypercube.

A five-dimensional, 16-node hypercube can perform the major classification calculations. The node processors have 0.5MFLOPS capablility, and the clustering algorithm is estimated to require 7.16MFLOPS.

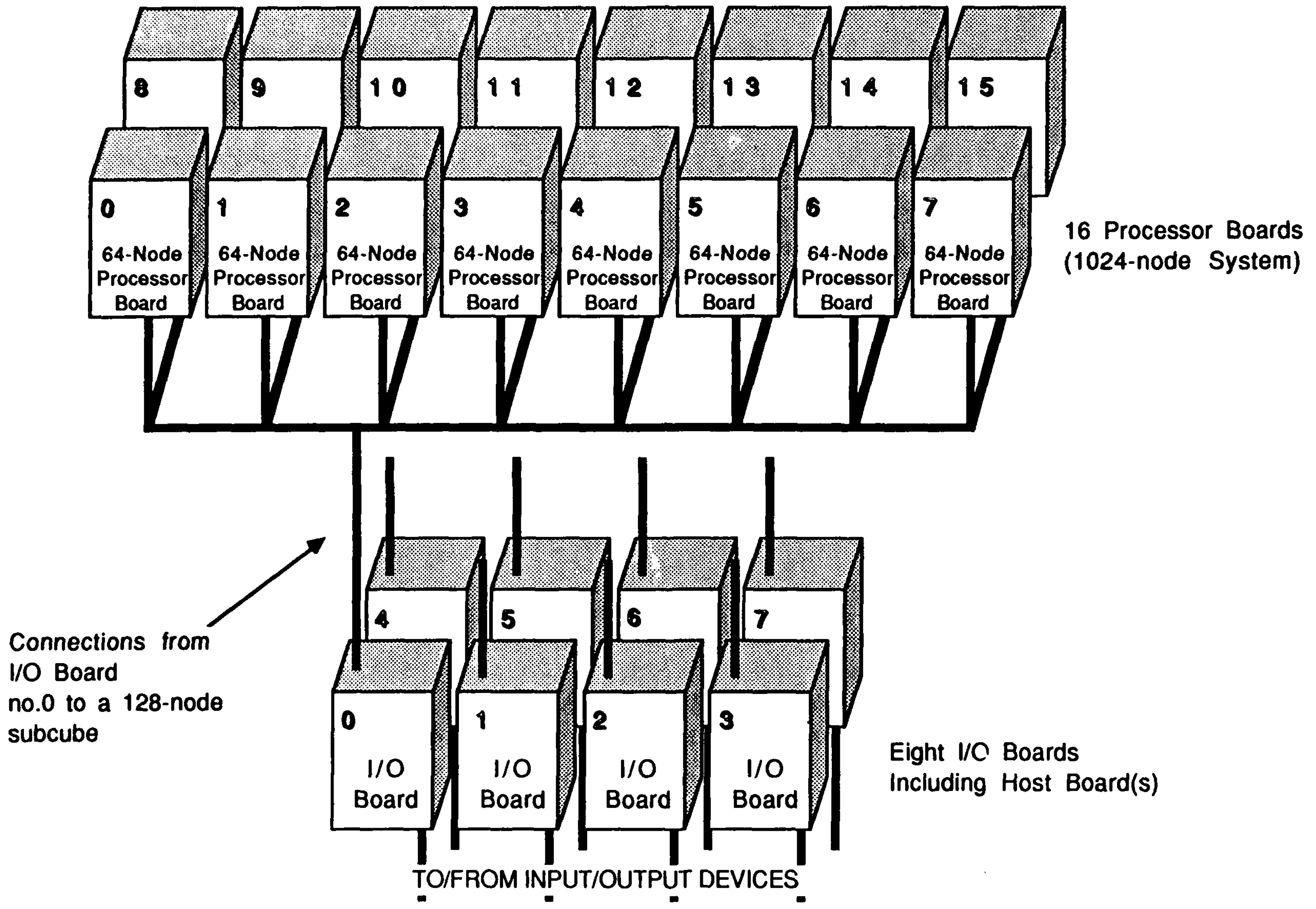
8
9
1 0
1 1
1 2
1 3
1 4
1 5
0 64-Node Processor Board
1 64-Node Processor Board
2 64-Node Processor Board
3 64-Node Processor Board
4 64-Node Processor Board
5 64-Node Processor Board
6 64-Node Processor Board
7 64-Node Processor Board
16 Processor Boards (1024-node System)
Connections from I/O Board no.0 to a 128-node subcube
4
5
6
7
0 I/O Board
1 I/O Board
2 I/O Board
3 I/O Board
Eight I/O Boards Including Host Board(s)
TO/FROM INPUT/OUTPUT DEVICES

Figure 13. Complete NCUBE10 System

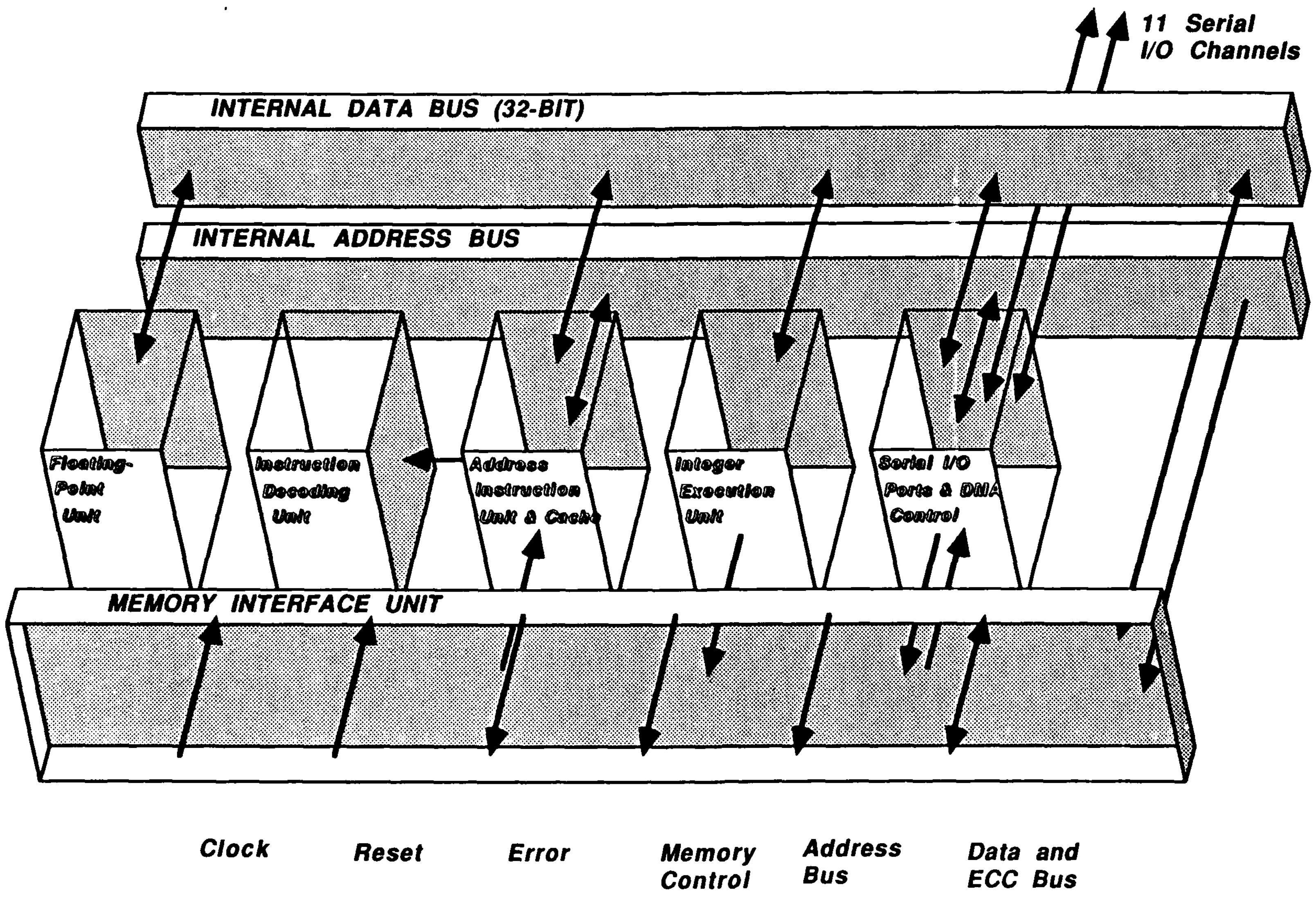

Figure 14. N-Cube Processor Chip Organization

Adding the 5% estimated for the LDS means that:

$$\left(\frac{7.16 + .05(7.16)}{0.5}\right) = 15^{+}$$

nodes processors are required.

This calculation assumes that data is always available for the processor nodes and that communication and I/O overhead is zero. The minimal system 64-node N-Cube7 has enough processing power to handle the distance and merging calculations as well as handle I/O overhead calculations and provide head room for expansion. The 64-node connection also allows the clustering algorithm to be conveniently mapped to the hardware without exceeding the 50 cluster maximum.

3.2.4 Force 68020 CPU Board Set

An evaluation of a group of standard CISC microprocessors communicating on a parallel bus interconnection topology is provided here. In this study, the Force central processing unit (CPU) boards were considered since they are high performance processors and Force is a reliable board supplier. In addition, the Force boards contain specific dedicated hardware that speeds message passing in a multi-processor system. The system would be structured as shown in Figure 15, 68020-based Classification System. The multiple processors communicate over a VME bus with each other and the data compression processor.

The multi-processor structuring of the classification algorithm would be similar to that for the transputers (see Section 3.2.2, Parsytec T800 Transputer), but with the tree structure implemented logically via message passing over the VME bus rather than by hardwired links. Each processor could load a program from a central memory or an on-board programmable read only memory (ROM).

3.2.4.1 Force Board Capabilities

Each processor would have a Motorola 68020/68882 processor with 256kBytes of local memory running at 20MHz and zero wait-states to provide a processing rate of 2MIPS and

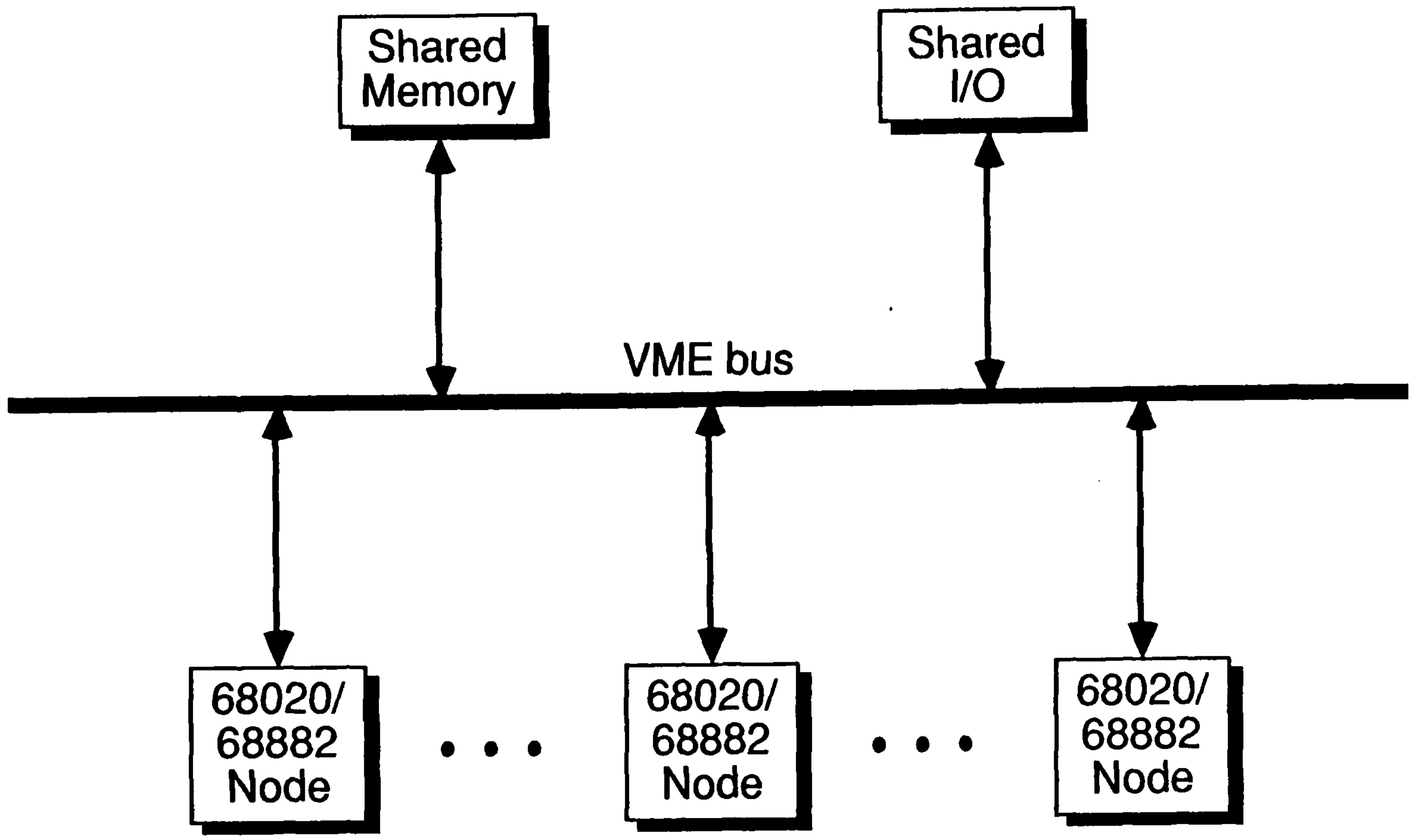

Figure 15. 68020-based Classification System

0.2MFLOPS per processor. Floating-point and integer operations may be overlapped subject to instruction dependencies. Dedicated message passing hardware allows a message, whose maximum length is eight bytes, to be broadcast to multiple processors with minimal latency. Additional resources on the board may be used for debugging (serial ports) or by real-time control software (timers).

The message passing hardware can broadcast a message to a maximum of 21 different CPU boards in parallel. A hardware first-in first-out memory (FIFO) allows a maximum of eight bytes to be sent in less than 20μsec.

3.2.4.2 Force Board Classification Performance

Multiple 68020/68882 CPUs will be required to perform the classification algorithm. The number of CPUs that would be required for a floating-point and fixed-point implementation of the classification algorithm is determined here. Given the much slower apparent floating-point performance of the CPUs (about 10 to 1), serious consideration must be given to a fixed-point implementation.

For single precision floating-point, the CPU can perform single additions, multiplications, and divisions in 3.5μsec, 4.5μsec, and 6.25μsec, respectively. For the combination of multiplication, division, addition, and subtraction operations in the classification algorithm, the throughput is 0.23MFLOPS. Alternatively, the CPU has 32-bit fixed-point operation times of 0.1μsec, 2.2μsec, and 3.9μsec for single additions, multiplications, and divisions. These times correspond to a throughput rate of 0.7MIPS for fixed-point arithmetic operations. This rate should be reduced by 30% or more to account for the additional range checking that must be performed in a fixed-point implementation. Therefore, for the mix of operations expected from the classification algorithm, the integer implementation would be about twice as fast as the floating-point implementation.

Communication among processors is flexible because the Force CPUs allow data to be broadcast to many processors in one communication cycle. Only 75 bytes need to be transmitted in the 68020 implementation. However, the message-passing mechanism of the Force boards prevents much of the communication from being overlapped with the arithmetic computation. All of these items add up to a 50% overhead to preserve real-time throughput rates with the Force board set.

Comparing these performance figures to the clustering algorithm processing requirement of 1.5(14.4)MFLOPS shows that 94 CPUs would be required for a floating-point implementation of the clustering algorithm alone. In other words, the CPU is not powerful enough to perform real-time calculations for even one cluster. An integer implementation would require about 43 CPUs for real-time execution rates. In addition, three extra CPUs would be used for the LDS and central control of the algorithm.

3.2.5 AISI AISI5000

An evaluation of a massively parallel group of fine-grained SIMD single-bit processors communicating by means of linear communication paths is provided here. The AISI is a SIMD machine organized as a linear array of up to 1024 single-bit processors and 32kbits of local memory (see Figure 3, AISI5000 System, and Figure 4, Organization of the AISI5000 Processor Elements). The AISI5000 was designed with a very large scale integration (VLSI) gate array—about 2000 gates. All of the AISI5000 processors operate in lockstep at a 7MHz clock rate. The organization of the AISI5000 processor allows it to be used not only for the image operations it was primarily designed for, but also for general purpose processing of vectors, matrices, and non-standard problems such as neural nets. Additional details of the AISI5000 processor can be found in Sections 2.2.1 and 2.4.2.

The AISI5000 is capable of:

- 1024 32-bit integer additions in 3.85μsec,
- 1024 32-bit integer multiplications in 318.75μsec, and
- 1024 32-bit integer divisions in 318.75μsec.

According to AISI, these times could be accurately extrapolated from the 8-bit addition and multiplication times by assuming a linear increase from 8 to 32 bits for addition and a geometric increase for multiplication. The time for division is assumed to be the same as for multiplication. This rough approximation becomes more accurate when more calculations are performed, as for vector operations, because the additional time for division is hidden in the time for overhead calculations.

Using the same line of reasoning as in Section 3.2.1, the time to calculate the clustering and merging equations with the AISI5000 for 5000 points and 50 clusters is:

$$50 \times 5000 \times \left(22\frac{15.41\mu\text{sec}}{1024} + 9\frac{319\mu\text{sec}}{1024} + 12\frac{319\mu\text{sec}}{1024}\right) = 1.72\,\text{seconds},$$

if 36-bit precision is used. The LDS calculations will also require a relatively small amount of time since they were estimated to require 0.3MFLOPS; this is 5% of the clustering and merging processing load. Also, AISI estimates that data I/O consumes about 5% of the processing power of the AISI5000. Therefore, we could increase the time by 5% to allow for the LDS calculations:

$$1.72\,\text{seconds} + 0.05 \times 1.72 = 1.81\,\text{seconds}$$

and then increase it by 5% to account for data I/O:

$$1.81\,\text{seconds} + 0.05 \times 1.81 = 1.90\,\text{seconds}.$$

This is a reasonable estimate of the entire processing time per frame including data I/O.

The AISI5000 cannot perform the classification operations at the required real-time rate and should not be considered for real-time execution of the REMIDS classification algorithms.

3.3 Classification Processor Acceptance Criteria

The first step of the evaluation process, explained in Section 1.2, is applied here for the classification hardware. The classification processor architecture recommended by this report should be the one most likely to produce a real-time demonstration within the limited development period available. The architecture may not be the highest performance processor nor the most suitable architecture since these attributes may make the processor difficult to program, purchase or interface. This section will define a set of criteria that allows objective comparison of different architectures and consistent rating of subjective features such as ease of programming.

The selected criteria are split into mandatory and desirable groups.

MANDATORY

- Bus system compatible with the data compression hardware,
- Established product,
- Adequate processing power, and
- Below size, weight, power and cost thresholds.

DESIRED

- Development support for parallelism (10)
- Head room (%) of basic system (9)
- Architecture I/O overhead (7)
- Expandability (7)
- Units in field (5)
- Field support (4)
- Cost (2)
- Size, weight, power (2)

In the desirable list, the development support for parallelism is considered to be the most important and is given a weight of 10. Many processors working together can perform a task in real-time if enough of them are used; the difficulty arises in programming them to work efficiently in parallel. Some development environments support parallelism well, while others simply provide the parallel capablility and leave management to the programmer. The programmer may also have difficulty utilizing the hardware to full capacity if the development system is inadequate.

Head room, defined as the percentage of spare processing power available in a baseline system, is given the next highest importance. Although additional processing power can be added in some cases by providing additional processors, this is not intended to be considered part of the head room value.

Expandability and architecture overhead were next in order of importance. Easy expandability may be critical if the classification algorithm requires more processing power than has been estimated in this report. Small increments are more desirable for economic reasons, but reducing the additional programming effort, required when expanding the processor size, should have a higher priority. Architecture overhead refers to the processing power required for operations that do not contribute to the processing of algorithm data, such as I/O management, processor synchronization, etc. Architecture overhead also includes processing power that is wasted when part of the hardware is unused because the hardware is not perfectly fitted to the algorithm. For instance, during synchronization of multiple parallel processors, some processors will idle—contributing nothing to performance—while waiting for other processors to complete their operations.

Next in importance were the number of units in the field and the available field support. The number of units in the field is used to estimate the maturity of the product. Normally, initial products are prone to difficulties that a more mature product would have previously eliminated. In the event of a problem with purchased hardware, the field support provided will determine how much time is lost before work can resume. While some companies provide spare boards for the user to stock on location or provide 24 hour replacement service, others may require that the failing board be shipped out for repair.

The final items were the size, weight, power consumption and cost. Within the maximum design constraints, these criteria were used to decide between two closely matched processors.

3.4 Classification Processor Analysis Results

Determining whether a processor possesses adequate processing power is a difficult criteria to evaluate. Besides raw peak processing rate (for example, the number of floating-point or fixed-point operations that can be processed per second), overhead such as I/O, processor synchronization and contention, etc., must also be considered in establishing accurate estimates of net throughput. Optimization techniques that could be applied in both hardware and software can provide significant performance gains, but they are difficult to predict.

Each of the potential classification processor architectures are evaluated using the criteria defined in Section 3.3. Table 4, Decision Analysis Worksheet (DAP 500/N-Cube); Table 5, Decision Analysis Worksheet (Parsytec/Force); and Table 6, Decision Analysis Worksheet (AISI5000), summarize the results of this evaluation for all of the previously described classification architectures. The following sections discuss the evaluation of each of the processors and derive the results presented in Tables 4–6.

3.4.1 AMT DAP 500

The second step of the evaluation process, explained in Section 1.2. is applied here to the DAP 500. The DAP 500 met all the mandatory criteria. A VME card cage, which is available for this system, facilitates communication to the data compression processor. Size, weight, power and cost for the DAP 500 are all acceptable. The product has been established as being readily available as well as being favored by the sponsor.

An analysis of the algorithm and its implementation on the DAP 500 showed the following. When using fixed-point arithmetic, the DAP 500 just barely performed the required operations in 1.5 seconds. The overhead, when using fixed-point arithmetic, was only 1%; this is drastically below the required value of 100% overhead. When using floating-point arithmetic, the DAP 500 took 1.74 seconds to perform the required operations; this is 16% longer than required and provides -16% head room. Because the 24-bit mantissa is smaller than a 36-bit fixed-point value, a shorter execution time might be expected. The difference is due primarily to the resynchronization overhead, incurred once for each floating-point normalization for any of the 1024 processors. Section 3.1.4 showed that 36-bit fixed-point calculations or 32-bit floating-point calculations are required for the classification equations. Because the DAP 500 cannot perform either of these calculations at the required rate, the DAP 500 should be removed from the list of systems being considered for the real-time implementation of the classification algorithms.

Table 4. Decision Analysis Worksheet (DAP 500/N-Cube7)

CLASSIFY OBJECTIVES		ALTERNATIVE EVALUATIONS					
		AMT's DAP500 (1024 PE's)			N-Cube's N-Cube7 (64 PE's)		
MUST (Mandatory, Measurable, Realistic)		INFORMATION	GO		INFORMATION	GO	
			NO	GO		NO	GO
Compatible Bus		VME		✓	SBX Connector (DR11-W)		✓
Established Product		YES		✓	YES		✓
Adequate Processing Power		NO (32-bit floating point)	✓		YES (32-bit floating point)		✓
		NO (32-bit fixed point)	✓				
		YES (16-bit fixed point)		✓			
Under Size, weight, power, cost		YES		✓	YES		✓
WANT (Desirable)	WT		SC	WT SC		SC	WT SC
Units In field	5	30 +	6	30	200 +	10	50
Field Support	4	On-Site Spares	10	40	Board Swap 24 Hours	8	32
Cost	2	$150,000 + $25,000 for a SUN	4	8	$158,000	4	8
Size (cubic feet), Weight (lbs.), Power (watts)	2	6,110,750	8	16	10, 250,1150	6	12
Expandability	7	None But Duplication	0	0	64-Node Increments	9	63
Development Support for Parallelism	10	Very Good	10	100	Very Good	10	100
Architecture Overhead Operating System I/O	7	Medium	7	49	Low	8	56
Headroom Percent without Expanding	9	1%	1	9	300%	10	90
TOTAL WEIGHTED SCORE			252			411	

Table 5. Decision Analysis Worksheet (Parsytec/Force)

CLASSIFY OBJECTIVES		ALTERNATIVE EVALUATIONS					
		Parsytec Transputer (13 PE's)			Force 68020 20MHz (46 PE's)		
MUST (Mandatory, Measurable, Realistic)		INFORMATION	GO		INFORMATION	GO	
			NO	GO		NO	GO
Compatible Bus		VME		√	VME		√
Established Product		YES		√	YES		√
Adequate Processing Power		YES (32-bit floating point)		√	YES (32-bit fixed-point)		√
Under Size, Weight, Power, Cost		YES		√	YES		√
WANT (Desirable)	WT		SC	WT SC		SC	WT SC
Units In field	5	100 +	9	45	20 +	5	25
Field Support	4	Board Swaps 3-4 Days - Stock Boards	6	24	Good	6	24
Cost	2	$56,000	10	20	$210,000	2	4
Size (cubic feet), Weight (lbs.), Power (watts)	2	1, 20, 50	10	20	6, 40, 325	7	14
Expandability	7	Configuration Software Changes	10	70	Prism Software	7	49
Development Support for Parallelism	10	Very Good	10	100	Poor	3	30
Architecture Overhead Operating System I/O	7	Low	10	70	High	3	21
Headroom Percent without Expanding	9	100%	9	81	100%	9	81
TOTAL WEIGHTED SCORE			430			248	

Table 6. Decision Analysis Worksheet (AISI 5000)

CLASSIFY OBJECTIVES		ALTERNATIVE EVALUATIONS					
		AISI-5000 (1024 PE's)					
MUST (Mandatory, Measurable, Realistic)		INFORMATION	GO		INFORMATION	GO	
			NO	GO		NO	GO
Compatible Bus		VME		✓			
Established Product		YES		✓			
Adequate Processing Power		NO (32-bit floating point)	✓				
		NO (32-bit fixed point)	✓				
		YES (16-bit fixed point)		✓			
Under Size, weight, power, cost		YES		✓			
WANT (Desirable)	WT		SC	WT SC		SC	WT SC
Units In field	5	40 +	7	35			
Field Support	4	???					
Cost	2	$70,000 + $25,000 for a SUN	6	12			
Size (cubic feet), Weight (lbs.), Power (watts)	2	4, 84, 530	8	16			
Expandability	7	128 PE Increments up to 1024	1	7			
Development Support for Parallelism	10	Poor	1	10			
Architecture Overhead Operating System I/O	7	Medium	8	56			
Headroom Percent without Expanding	9	0%	1	9			
TOTAL WEIGHTED SCORE			145+				

The weighted criteria for the DAP were given the following scores. There are currently 25–30 third generation DAPs in the field. Over the past eight years, 20 first- and second-generation machines have been delivered. Compared with the other systems, this number of units in the field is small and therefore a score of 6 is given. Field support is rated at 10 out of a possible 10 points, because spares could be made available for immediate replacement for a price that is reasonable when compared to the cost of the total system.

The score for the cost of the system is 4 because more powerful systems can be purchased for a significantly lower cost. The size, weight and power consumption are relatively small; therefore, the DAP500 is given a high score of 8.

A score of 0 is given for the expandability criteria. Currently no modularity is designed into the DAP500 processor. More processing power can be provided only by duplicating the system. AMT has just begun manufacturing a DAP600 series processor with a 64×64 array. The cost for a complete system is $340,000 – $350,000. Although the cost is twice the cost of the DAP500, the system has four times the number of processors and an increased I/O rate.

A score of 10 is given for the development support for parallelism criteria. A large library of standard routines is available and has been optimized in assembly code for the DAP. Fortran Plus allows code to be written that is more readable because the details of the hardware are transparent to the programmer. Assembly code allows the hardware to be programmed for optimum use of resources when processing time is critical.

The architecture overhead associated with the DAP500, when compared to the other systems, is in the middle range. Data I/O overhead is estimated to consume about 8% of the processing power of the system; the DAP500 received a relatively high score of 7 for this criteria.

The composite score for the DAP500 is 243. It is ranked in 3rd place, behind the Parsytec Transputer and the N-Cube Hypercube systems.

3.4.2 Parsytec T800 Transputer

The second step of the evaluation process, explained in Section 1.2, is applied here to the Parsytec Transputer. The transputer multi-processor meets all of the required criteria for performing the real-time REMIDS classification algorithm. A VME interface card allows communication with the rest of the system. The system size, weight and cost are within acceptable limits. Of the architectures considered, the transputers have a performance-per-cost ratio that is one of the best. Parsytec is also a well established company with world markets and has shipped well over 100 transputer boards to customers.

Given the additional programming effort required and the discussions in Sections 3.1.4 and 3.2.2.2 regarding fixed-point versus floating-point, a floating-point implementation of the algorithm would be preferrable to a fixed-point implementation. The floating-point implementation requires 11 processors and the fixed-point requires four less, but the additional programming cost could easily exceed the $13,000 hardware savings. In addition, the implementation risk expands greatly since the fixed-point implementation would be much less robust with respect to input data values and would possibly be less precise.

The composite score for the Parsytec Transputer is 430, which ranks this architecture in 1st place.

3.4.3 N-Cube N-Cube7

The second step of the evaluation process, explained in Section 1.2, is applied here to the N-Cube Hypercube. The N-Cube met all the mandatory criteria. An N-Cube system with the minimal configuration of 64 nodes has more than enough processing power for the classification algorithms. The system is easily upgradable to twice the number of nodes and with the addition of a card cage may be upgraded to 1024 nodes. The data I/O bandwidth to and from the classification processor can be met with the serial host interface and is capable of communication with the data compression hardware if an additional board is bought for the data compression card cage.

The desirable criteria scores are as follows. The N-Cube processor is well established, as is the Hypercube concept, and N-Cube currently has 200 units in the field; this score is 10. The field support score is 8. The N-Cube comes with a 1 year warranty. Field support consists of exchanging a defective board for a replacement within 24 hours. The cost score of 4 was based on a $157,500 price. The size, weight and power score of 6 was based on a 10 ft^3, 250 lb, 1150 watt system. The expandability was determined to be good and rated a score of 9. Development support was rated very good and scored 10. Architecture overhead was accepted as low and scored an 8. The head room for a minimal configuration 64-node system is 300% and ranked a score of 10.

The composite score for the N-Cube Hypercube is 411, which ranks this architecture in 2nd place behind the Parsytec Transputer system.

3.4.4 Force 68020 Board Set

The second step of the evaluation process, explained in Section 1.2, is applied here to the Force 68020 board set. Based on the performance evaluation in Section 3.2.4, the 68020/68882, using floating-point arithmetic, cannot be seriously considered for a real-time implementation of the REMIDS classification algorithm. Using fixed-point arithmetic, 46 CPUs could be configured together, but the Force board limit of 21 CPUs in a broadcast operation would make coordinating the algorithm execution difficult and would increase the data traffic on the VME bus. In addition, the programming effort to control such a large multi-processor would be prohibitive.

The composite score for the Force 68020 board set is 248, which ranks the architecture in 4th place behind the Transputer, Hypercube and DAP systems.

3.4.5 AISI AISI5000

The second step of the evaluation process, explained in Section 1.2, is applied here to thc AISI AISI5000. The AISI5000 met all the mandatory criteria. The VME interface is compatible with a large variety of hardware so interfacing to the data compression hardware is not a problem. AISI is an established company with 40 machines in the field. The size, weight, power and cost thresholds were all met comfortably.

Because Section 3.2.1.4 showed that 36-bit fixed-point calculations are required for the classification operations and the AISI5000 is incapable of processing at real-time rates with this precision, the AISI5000 should be removed from the list of systems being considered for the real-time implementation of the classification algorithms.

Although the AISI5000 is expandable in increments of 128 PEs, the full 1024 PE system is used as the baseline. No additional increments are possible with the exception of duplication and next generation hardware.

The development support for parallelism is poor. Although the system does handle parallelism well and has an extensive number of commands and library routines for vectors, the system works primarily on image data. The 8-bit image data commands are not appropriate for the floating-point computations that must be performed.

The architecture overhead for data I/O is about 5%; this is a middle range score compared with the other architectures considered.

The composite score for the AISI5000 is 145, which ranks the architecture in 5th place.

4.0 PROPOSED HARDWARE CONFIGURATION

An overall system configuration for the REMIDS II Airborne processor is depicted in Figure 16, REMIDS II Airborne Processor. At a high level, the two major processor components, data compression and classification, are arranged in a configuration with a separate Host CPU acting as a common system controller for each bus. This allows each processor to operate independently with its bus and to link with the host when data exchanges are required. With their heavier processing loads, the data compression and classification processor buses are never directly coupled, thus substantially avoiding the inevitable contention delays. The proposed data compression hardware consists of programmable pipeline modules produced by Datacube. The proposed classification hardware consists of several T800 Transputers that have been incorporated with additional memory into processing nodes by Parsytec.

4.1 Operational Sequencing

The operational sequencing of the airborne processor is summarized below. The inherent parallel operations are described in a sequential manner for clarity.

- The host CPU initializes data compression and classification processors and other components of the system.
- The scanner provides the next line of the 3-channel image data that is formatted by the scanner interface.
- The data compression processor adds a new scan line to its input frame buffer. If the full frame is assembled, processing begins for this frame. After frame processing is complete, candidate target coordinates and pixel values are transferred to the host CPU where they are placed into shared RAM for access by the classification processor (see Figure 17, Execution Sequence Through the Data Compression Processor).
- The classification processor accepts candidate target information from the host RAM and performs clustering and LDS algorithms. Pixel locations where mines are detected are compiled and transferred to shared host RAM for access by the data compression processor.
- Detected mine locations are accepted by the data compression processor. They are then used to assemble an image that is superimposed upon the original source image from which the detected mines were derived. The source images

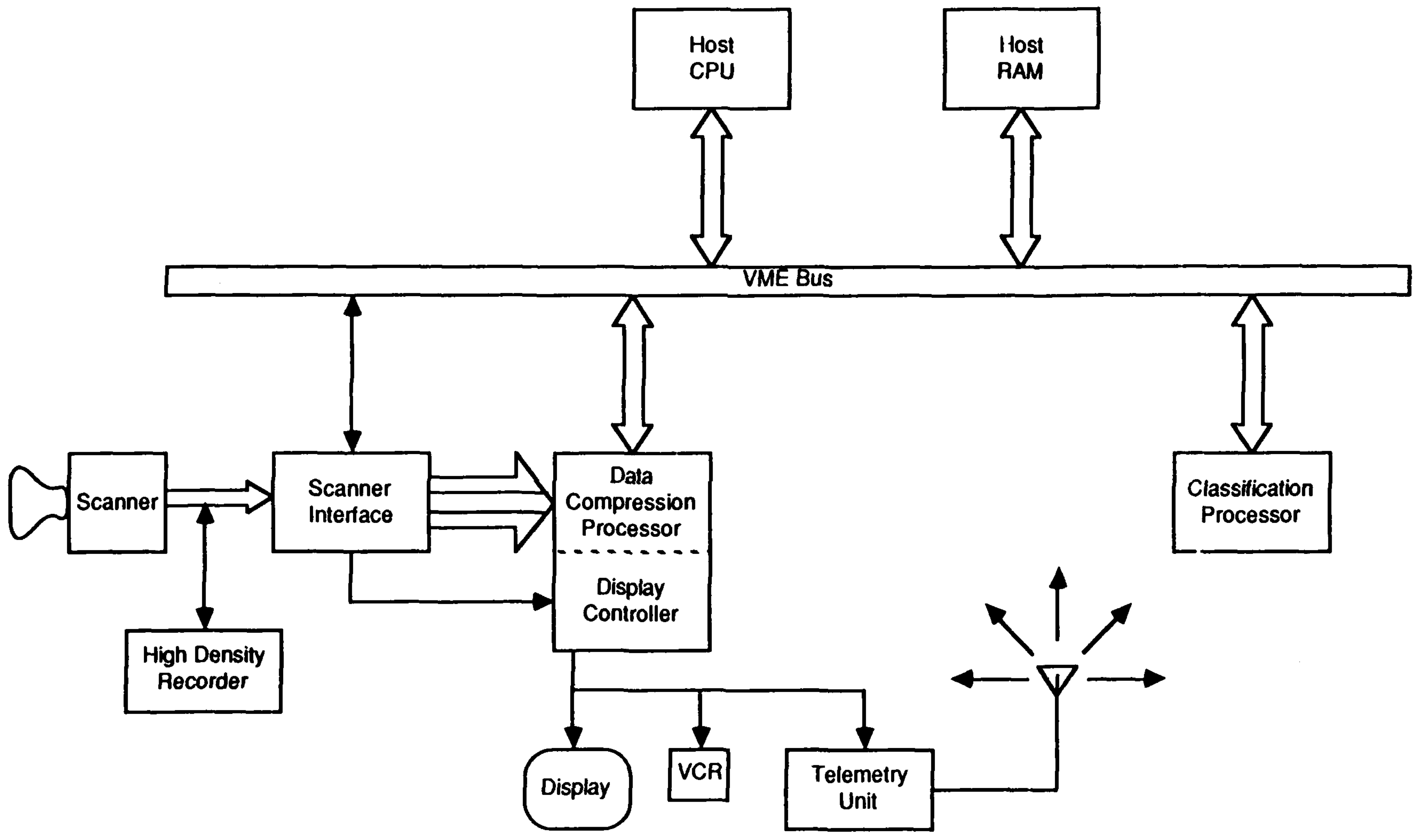

Figure 16. REMIDS II Airborne Processor

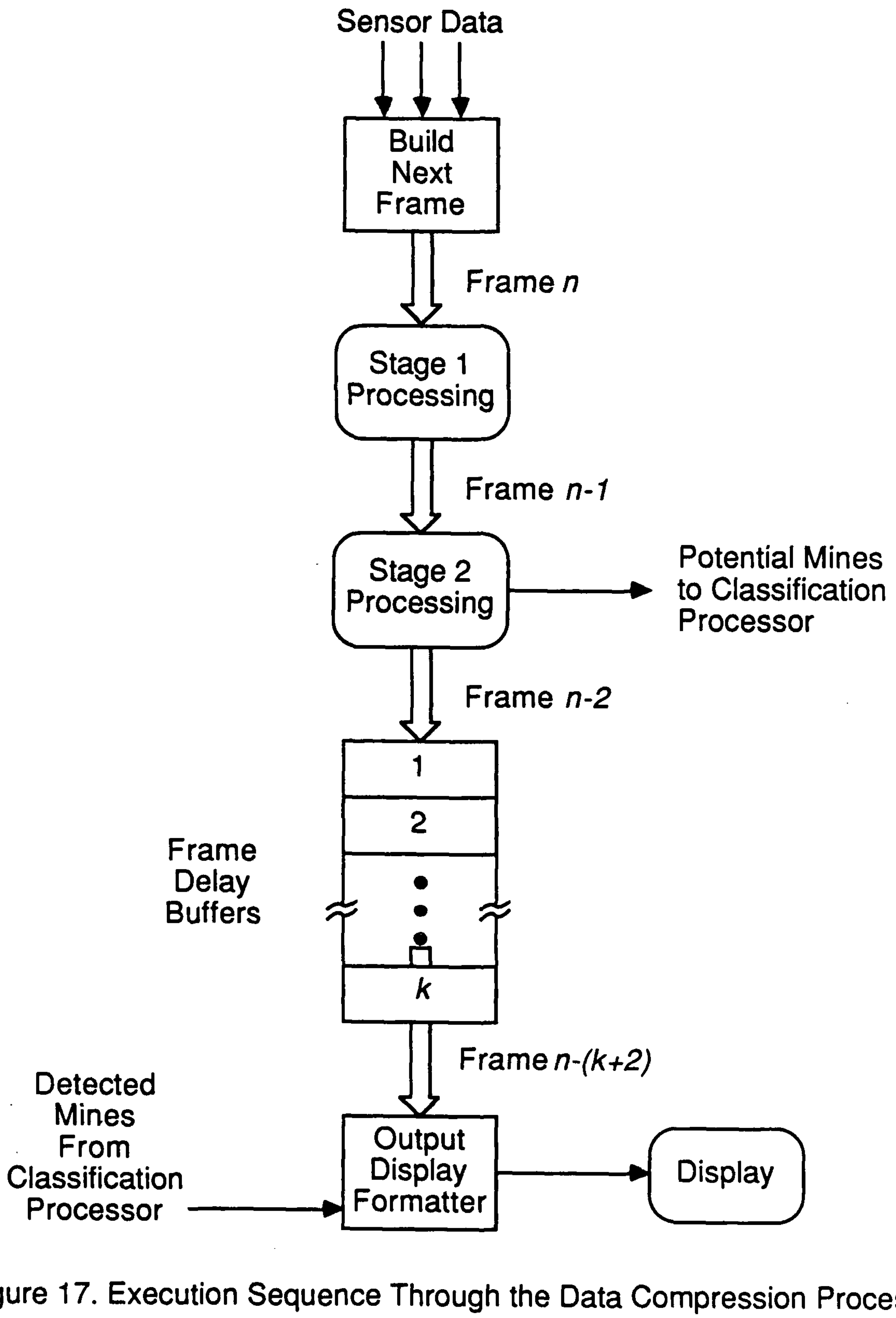

Figure 17. Execution Sequence Through the Data Compression Processor

are buffered within the data compression processor and are aligned with the detected mine masks based upon the overall processing latency through the image processing chain.

- The scanner image with designated mines is converted to a red, green, blue (RGB) format and displayed on the high-resolution monitor with suitable along-track compression. The uncompressed video frame is recorded on the video cassette recorder (VCR) and accepted by the telemetry unit for transmission to the ground observation station.

4.2 Open Issues

Further development is required in several areas of the operational system design. The following issues remain unresolved:

- Operator interaction during data collection and airborne processing.
- Recovery from data-dependent errors that occur within the classification processor, such as:
  - Too many clusters, and
  - Too low a threshold.
- Deletion of old data from classification clusters.
- Detailed bandwidth analysis.
- Telemetry and recorder interface overhead effects.